Solving Partial Differential Equations using Radial Basis Functions

Chengxi Zeng, Andreas Michael, Celia Tugores-Bonilla, Natasha Moore, Jules Boissier
Alberto Gambaruto
Department of Mechanical Engineering, University of Bristol

23rd October 2021

Declaration

The accompanying Group Industrial Project report entitled: "Solving Partial Differential Equations using Radial Basis Functions" is submitted in the fourth year of study towards an application for the degree of Master of Engineering in Mechanical Engineering at the University of Bristol. The report is based upon independent work by the candidates. All contributions from others have been acknowledged above. The supervisor is identified at the start of the report. The views expressed within the report are those of the authors and not of the University of Bristol.

We hereby declare that the above statements are true:

AM	[illegible]	[illegible]	NMoore	[illegible]
Andreas Michael	Celia Tugores-Bonilla	Simon Zeng	Natasha Moore	Jules Boissier

Copyright

Work Allocation

	Andreas Michael	Celia Tugores	Simon Zeng	Natasha Moore	Jules Boissier
1. Introduction	✗	✗			
1.1 Previous work	✗		✗		
1.2 Aims and Objectives	✗	✗	✗	✗	✗
2. Methodology					
2.1 Radial Basis Functions	✗				
2.2 RBFs for PDEs	✗			✗	
3. Applications					
3.1 Linear PDEs				✗	✗
3.2 Function Reconstruction			✗		
3.3 Methods for least squares minimisation for non linear problems		✗			
3.4 Non linear PDEs: example using Burgers' Equation	✗	✗			
4. Results and Discussion					
4.1 Linear PDEs: the Poisson Equation				✗	
4.2 Linear PDEs: the Heat Equation				✗	
4.3 Linear PDEs: the Wave Equation					✗
4.4 Function Reconstruction			✗		
4.5 Non linear PDEs: Burgers' Equation	✗	✗			
5. Conclusions		✗			
Signature	[illegible]	[illegible]	[illegible]	NMoore	[illegible]

Mitigation Plan

Event/Issue	**Potential/actual impact on project**	**Actions(s) taken to mitigate impact on project outcomes**	**Potential/ Remaining Impact**
Inability to work together and increased difficulty in communication.	Less support within the group and a decreased awareness of issues and contributions of individuals.	Frequent videoconferencing and instant messaging.	Potential cohesion issues within the report.
Some of the group personally affected by the virus so they were isolating and unable to work for 14 days.	Reduction of the individual's contributions and inability to work smoothly as a group.	Extra efforts were put by affected individuals later on and peer support through isolation periods.	Potential reduction of the scope of the project.
Faced some connectivity problems and increased difficulty on working under such conditions.	Difficulty to work, communicate and receive feedback.	Peer support.	No significant impact to the final report.

Abstract

This report uses Radial Basis Function (RBF) based methods to discretise standard Partial Differential Equations (PDEs), aiming to assess their performance compared to the analytical results and alternative discretisation methods. All methods used are based on either global or local support RBF interpolation. They include Kansa's method, the Radial Basis Function-Finite Difference method (RBF-FD) and the Local Radial Basis Function Collocation method (LRBFCM). Schemes are created to solve linear equations such as the Poisson, heat and wave equations and the non linear Burgers' equation. Time derivatives are discretised using either an explicit or implicit scheme, and least squares solvers are used to provide results for the implicitly discretised Burgers' equation. Results provided for the linear equations are compared to analytical solutions and show the accuracy and versatility of the method, which is mesh free. The explicit methods for the heat and wave equation are shown to be accurate provided an appropriately small time step is taken. The implicit methods give less accurate results, due to issues with ill-conditioned matrices, but are more stable for larger time steps. Solutions to the Burgers' equation give low errors, $\varepsilon = L_\infty = 0.0011$, and a comparable performance of the RBF based methods to other numerical methods. The implicit time discretisation deals with the stabilisation of the solution to the Burgers' equation at high Re numbers. Applications are provided in the field of image reconstruction where surfaces are reconstructed using a solution to the Poisson equation provided through the global Kansa RBF scheme. A comparison of this method to RBF implicits and the Hermite RBF method shows it provides higher accuracy results coupled with computational inefficiencies. The RBF methods shown are versatile and provide accurate results for all PDEs examined in this report.

Contents

1 Introduction **8**
1.1 Previous work . . . 9
1.2 Aims and Objectives . . . 9

2 Methodology **10**
2.1 Radial Basis Functions . . . 10
2.1.1 Global Support Radial Basis Functions . . . 11
2.1.2 Local Support Radial Basis Functions . . . 12
2.2 RBFs for PDEs . . . 13
2.2.1 Kansa's Method . . . 13
2.2.2 RBF Finite Difference . . . 14
2.2.3 Local Radial Basis Function Collocation Method . . . 15

3 Applications **16**
3.1 Linear PDEs . . . 16
3.2 Function Reconstruction . . . 17
3.3 Methods for least squares minimisation for non linear problems . . . 18
3.3.1 The Gauss-Newton Method . . . 19
3.3.2 The Levenberg-Marquadt Method . . . 19
3.4 Non linear PDEs: example using Burgers' Equation . . . 21
3.4.1 Explicit Time Scheme and Global RBF . . . 21
3.4.2 Explicit Time Scheme and Local RBF . . . 22
3.4.3 Explicit Time Scheme and RBF-FD . . . 22
3.4.4 Implicit Time Scheme and Global RBF . . . 23
3.4.5 Implicit Time Scheme and RBF-FD . . . 24

4 Results and Discussion **26**
4.1 Linear PDEs: the Poisson Equation . . . 26
4.1.1 Kansa's Method . . . 26
4.1.2 RBF-FD . . . 27
4.2 Linear PDEs: the Heat Equation . . . 29
4.2.1 Explicit Time Scheme . . . 30
4.2.2 Implicit Time Scheme . . . 30
4.3 Linear PDEs: the Wave Equation . . . 32
4.3.1 Explicit Time Scheme . . . 33
4.3.2 Implicit Time Scheme . . . 33
4.4 Function Reconstruction . . . 34
4.4.1 3D Surface Reconstruction . . . 34
4.4.2 2D Boundary Reconstruction . . . 35
4.5 Non linear PDEs: Burgers' Equation . . . 37
4.5.1 Non linear least squares methods . . . 37
4.5.2 1D Burgers' Equation . . . 38
4.5.3 2D Burgers' Equation . . . 40

5 Conclusions **44**

Glossary

ChSC Chebyshev Spectral Collocation. 38, 40

EG-RBF Explicit Global Radial Basis Function scheme. 39, 41, 42, 43

EL-RBF Explicit Local Radial Basis Function scheme. 39, 40, 41, 42

ERBF-FD Explicit Radial Basis Function- Finite Difference scheme. 39, 41, 42

FD Finite Difference. 8, 43, 44

G-N Gauss-Newton. 19, 20, 37

HRBF Hermite Radial Basis Function. 10, 34, 35, 36, 44, 45

IG-RBF Implicit Global Radial Basis Function scheme. 39, 40, 41, 42, 43, 44

IRBF-FD Implicit Radial Basis Function- Finite Difference scheme. 39, 41, 42, 43, 44, 45

L-M Levenberg-Marquadt. 19, 37, 38

LRBFCM Local Radial Basis Function Collocation Method. 5, 9, 10, 13, 15, 16, 21, 22, 42, 43, 44

MOL Method of Lines. 38, 40

MQ Hardy Multiquadrics. 9, 10, 40

PDE Partial Differential Equation. 5, 6, 8, 9, 10, 13, 14, 15, 16, 18, 21, 26, 29, 32, 37, 38, 44, 45

PS Pseudospectral methods. 8

RBF Radial Basis Function. 5, 6, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 21, 22, 23, 24, 27, 29, 33, 34, 35, 36, 38, 39, 40, 41, 42, 43, 44, 45, 46

RBF-FD Radial Basis Function-Finite Difference. 5, 6, 8, 9, 10, 13, 15, 16, 17, 21, 22, 24, 27, 29, 31, 33, 37, 39, 41, 42, 43, 44, 45

Re Reynolds Number. 5, 21, 38, 39, 40, 41, 42, 43

TPS Thin Plate Splines. 9, 10

1 Introduction

Differential equations are the basis for understanding a wide range of physical phenomena arising in fluid mechanics, geosciences, biology or economics amongst other disciplines. They play an important role in virtually modelling every physical and biological interaction of the real world. However, many differential equations are not directly solvable, as they do not have a closed-form expression, and their solution is approximated by numerical solutions instead.

The numerical solution of differential equations has been studied for over a century. The first method proposed was the finite difference (FD) method, introduced by Richardson (1911), who used Taylor series expansions to approximate the derivatives. Generally, the method is easy to implement and has been a dominant methodology ever since. Pseudospectral methods were introduced later on, in the early 1970's for simple geometries by Gary (1970), which offered higher accuracy and computational efficiency. Moreover, Kansa (1990) was the first to use radial basis functions (RBFs) as a tool to discretise spatial derivatives in PDEs, which offered a novel approach to their numerical solution. The method can be seen as a generalised pseudospectral method which offered geometric freedom and node ordering independence whilst keeping spectral accuracy.

Ideally, the approximation to the solution of a PDE should be high-order accurate, computationally efficient, flexible regarding the geometry and easily implemented. Nevertheless, commonly used methods fail to fulfil all the mentioned requirements. Finite difference methods can be high-order accurate, but they require a structured grid. Pseuspectral (PS) methods have shown to be even more accurate but the method suffers severe geometric limitations and depends on regular node layouts. In the Fourier case, they even require periodic boundary conditions. Other methods such as finite element methods are remarkably flexible but it is hard to achieve high accuracy and mesh generation also becomes increasingly difficult at higher spatial dimensions. Radial basis functions depend on the Euclidian distance from a centre point $\boldsymbol{x}_i$: $\phi(||\boldsymbol{x} - \boldsymbol{x}_i||)$, and a shape factor, c, is generally included. RBFs do not require a mesh or grid to discretise derivatives, which allows for geometric flexibility and local refinements where required, hence the computational cost does not depend on the complexity of the geometry. Moreover, working in higher dimensions does not increase the difficulty of the method, as opposed to finite element methods, which results in an ease of implementation. Despite the high accuracy of the method described by Kansa, computational cost and the scalability to large computer systems remained lingering concerns [Fornberg & Flyer (2015)]. Nevertheless, different methods using RBFs have been explored and it has been recently discovered that using RBFs for generalised FD methods could offer the ideal accurate, cost effective and flexible solution to the numerical solution of differential equations [Tolstykh & Shirobokov (2003)]. RBF-FD methods offer high computational speeds, high accuracy levels compared to PS and Kansa's method, and opportunities for adaptive refinement and large-scale parallel computing. The use of RBFs for the solution of differential equations has proven to be highly successful against other approaches, demonstrating high feasibility and cost advantages.

For this reason, this report explores the use of radial basis functions as a tool to discretise spatial derivatives and solve different partial differential equations for various applications. The research considers different RBF schemes, such as the global Kansa RBF method, a local RBF method, the Hermite RBF method and the RBF-FD method, as well as different time scheme discretisations.

1.1 Previous work

Radial Basis Functions are radial functions used for multivariate interpolation, where data site inputs are used to create a univariate function, $f : R^n \to R$ [Fornberg & Flyer (2015)]. The first use of RBFs was by Hardy (1971) who used multiquadrics (MQ) to represent 2D topographic maps using scattered data and is related to kriging which is based on the work from Krige (1951). Following the discovery and use of multiquadrics and, subsequently, Thin Plate Splines (TPS) by Duchon (1977), RBFs became prominent in applications for scattered data interpolation (mesh free). Developments in compactly supported RBFs were done by Wendland (1995), to deal with ill-conditioned interpolation systems.

Conventionally, mesh-based techniques such as the Finite Element method [Le Méhauté (1990)] and Finite Difference methods [Xu & Lu (1988)] were used for image reconstruction. However, RBF implicit surface reconstruction was firstly proposed by Carr *et al.* (1997) for medical images. Initially, the size of the problem treated was constrained to computational power, however, this issue was overcome with advancing technology. Kazhdan *et al.* (2006) proposed solving the Poisson equation as a solution for surface reconstruction which creates a smooth watertight surface and has noise prevention advantages. Another approach for surface reconstruction is studied by Macêdo *et al.* (2011), where Hermite data is used for interpolation.

Additionally, RBFs were first used by Kansa (1990) to create a discretisation scheme for the spatial derivatives in a PDE. The method is called Kansa RBF collocation method. It was used extensively to discretise and solve PDEs such as the linear advection-diffusion equation and the elliptic Poisson equation by the original work from Kansa (1990). Following the work from Kansa (1990), a local Radial Basis Function-Generated Finite Difference (RBF-FD) method was developed and introduced by Tolstykh & Shirobokov (2003) and Shu & Yeo (2003). A multitude of applications followed including solving PDEs on curved surfaces and over 3D bodies [Piret (2012)]; modelling turbulent vortical flame propagation by Kansa *et al.* (2009); radiative transfer by Kindelan *et al.* (2010); and finally, extensive use in the fluid mechanics field such as in simulations of water flow by Wong *et al.* (1998) or the solution of the Navier-Stokes equations by Demirkaya *et al.* (2008). RBF derived methods were also used to discretise non linear PDEs including the numerical solution of the Burgers' equation in 1D by Khater *et al.* (2008) and Ali *et al.* (2011), and 2D by Ali & Haq (2009) and Sarler *et al.* (2012).

1.2 Aims and Objectives

The aim of this group industrial project is to apply and assess the performance of different RBF based methods, such as Kansa, RBF-FD, LRBFCM or Hermite, to solve partial differential equations for different applications. The applications detailed in this research include the use of RBFs on image processing, the solution of linear PDEs, such as the Poisson, heat and wave equation, and non linear PDEs, such as the Burgers' equation, all using RBF based discretisation methods.

For the application to linear PDEs the aim is to solve the Poisson, heat and wave equation for two dimensional cases. The accuracy is then assessed against analytical solutions. For the equations that have a time derivative (heat and wave in this case) both explicit and implicit methods are tested for the approximation in time.

Moreover, radial basis functions are employed to create implicit functions for surface reconstruction in three different ways: RBF implicits, Hermite Radial Basis Function

(HRBF) implicits and Poisson Surface Reconstruction. Three approaches are designed to be compared in multiple experiments both in 2D and 3D regarding their accuracy and computational performance. Some real life applications are demonstrated using different techniques that show their performances and feasibilities.

Regarding the non linear PDEs application, the aim is to solve the 1D and 2D Burgers' equation numerically through five different discretisation schemes that use RBF methods (Kansa, RBF-FD and LRBFCM) to discretise spatial derivatives and either an Euler explicit or implicit discretisation scheme for the time derivatives. Therefore, the main objectives of the section are firstly, to present a numerical solution of both the 1D and 2D Burgers' equations; secondly to perform convergence studies on all the methods; thirdly to assess the performance of RBFs in comparison to each other, the analytical solution and to different methods and schemes from prior research on the Burgers' equation; and lastly, to study the performance and convergence of different non linear least squares methods algorithms for the implicit time schemes used to solve the PDE.

2 Methodology

2.1 Radial Basis Functions

Radial Basis Functions are functions used to map multivariate scattered data from a specific amount of dimensions in Euclidean space to a function $\Phi : R^n \to R$. The function is called radial if there exists a univariate function $\phi : [0, \infty) \to R$ such that the following relationship holds true [Fasshauer (2007)]:

$$\Phi(\boldsymbol{x}) = \phi(r), \qquad r = \|\boldsymbol{x}\| \tag{1}$$

where $\|\cdot\|$ is the Euclidean norm, $\phi(r)$ is the univariate basic function which generates the basis functions $\Phi(\|\boldsymbol{x} - \boldsymbol{x}_i\|)$ by shifting to different centres $\boldsymbol{x}_i \in R^n$. Hence all basis functions $\Phi(\|\boldsymbol{x} - \boldsymbol{x}_i\|)$ are radially symmetric about their centres $\boldsymbol{x}_i$ and are called Radial Basis Functions.

Radial basis functions are used to interpolate large scattered data sets, hence they do not require a mesh to be fitted. They are better suited to deal with complex geometries and large sets of data in comparison to other functions such as univariate polynomials [Fasshauer (2007)]. A univariate basic function $\phi(r)$ is invariant to complexities added by additional dimensions in the original data sites. This allows for a large number of dimensions to be used, as the interpolation with RBFs does not increase in complexity with the dimension number. The following are common basic function choices used to form the RBFs:

- Hardy Multiquadrics (MQ) [Hardy (1971)]: $\phi(r) = \sqrt{r^2 + c^2}$
- Gaussian: $\phi(r) = \exp(-c^2 r^2)$
- Thin Plate Splines (TPS) [Duchon (1977)]: $\phi(r) = r^{2n} log(r)$

where c is the shape parameter and n an integer depicting the order of the TPS. For this work it was set to $c = \sqrt{5/N}$, where N is the number of data sites used, following a brief numerical testing. For all applications presented in this report, the RBFs chosen are the Hardy Multiquadrics (MQ) as they have proven to be the most accurate for scattered data interpolation by Franke (1982). Global and local support RBFs are presented in the sections to follow, in the context of interpolation.

2.1.1 Global Support Radial Basis Functions

Interpolation is the fitting of a continuous function, $f : R^n \to R$ to given data values $s(\boldsymbol{x}_i)$ at given data sites $\boldsymbol{x_i}$, where $\boldsymbol{x_i} \in R^n$, such that $f(\boldsymbol{x_i}) = s(\boldsymbol{x_i})$. This is called multivariate data interpolation. Data can be scattered, as a mesh is not required to explicitly define basic functions. The function can provide required information by being evaluated at various locations in the domain, not involving the input points. This problem is solved by assuming a linear combination of radial basis functions to the interpolant, f:

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \lambda_i \phi(\|\boldsymbol{x} - \boldsymbol{x}_i\|), \qquad \boldsymbol{x} \in R^n \tag{2}$$

where $\boldsymbol{x}$ is the free variable of positions for the function to be evaluated at, $\boldsymbol{x}_i$ are the given data sites with a known function value, $i = 1, ..., N$, $\lambda_i = [\lambda_1, ..., \lambda_N]$ are scalar coefficients facilitating the function fit and $\phi(\|\boldsymbol{x} - \boldsymbol{x_i}\|)$ is the radial basis function.

The assumption from Eqn. (2) is used to form a system of linear equations to be solved for the coefficients, λ_i:

$$\begin{pmatrix} \phi(\|\boldsymbol{x}_1 - \boldsymbol{x}_1\|) & \phi(\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|) & \dots & \phi(\|\boldsymbol{x}_1 - \boldsymbol{x}_N\|) \\ \phi(\|\boldsymbol{x}_2 - \boldsymbol{x}_1\|) & \phi(\|\boldsymbol{x}_2 - \boldsymbol{x}_2\|) & \dots & \phi(\|\boldsymbol{x}_2 - \boldsymbol{x}_N\|) \\ \vdots & \vdots & & \vdots \\ \phi(\|\boldsymbol{x}_N - \boldsymbol{x}_1\|) & \phi(\|\boldsymbol{x}_N - \boldsymbol{x}_2\|) & \dots & \phi(\|\boldsymbol{x}_N - \boldsymbol{x}_N\|) \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_N \end{pmatrix} = \begin{pmatrix} s(\boldsymbol{x}_1) \\ s(\boldsymbol{x}_2) \\ \vdots \\ s(\boldsymbol{x}_N) \end{pmatrix} \tag{3}$$

where $s(\boldsymbol{x}_i)$ are the known values of the function at the input data sites. This linear system, $A\lambda = s$, has a solution if A is non-singular and the problem is well posed. Based on the Mairhuber-Curtis theorem [Fasshauer (2007)], an interpolation problem does not ensure a solution if a predefined function is chosen for the scattered data set. RBFs are inherently undefined (Eqn. (1)), since the basic function generates radial basis functions by shifting centres. Additionally, with the use of multiquadrics (MQ), the RBF matrix is invertible as proven by Micchelli (1984), and unisolvency is ensured.

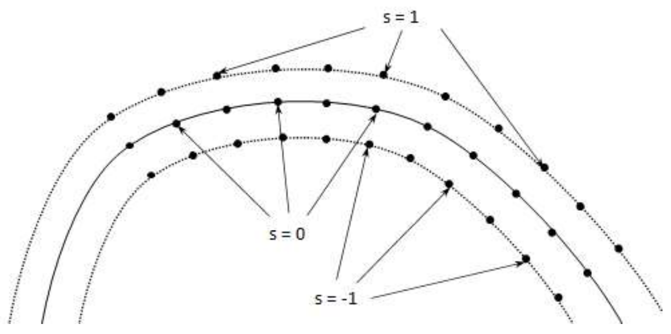
Linear systems can be directly solved if the matrix, A, is well conditioned and small. However with globally supported RBF interpolation problems, such as the previously imposed problem, the RBF matrix is usually densely populated and has a high condition number due to the large number of data site inputs ($\boldsymbol{x}_i$). Results are prone to error if such a system is directly solved. Options to minimise errors include decomposing the matrix through singular value decomposition, LU decomposition amongst others, or using iterative solvers such as the conjugate gradient method or the generalised minimum residual method (both available in MATLAB through `pcg` and `gmres`).

An example of scattered data interpolation for surface reconstruction and morphing is presented. A function is created which is expressed such that at the surface its value is $f = 0$. Known points on the surface are provided and interpolation gives information at points not included in the input set. Hence, it can be used to reconstruct a surface or 3D object as done by Carr *et al.* (2001) or create a morphing sequence between an initial and final surface as done by Turk & O'Brien (1999).

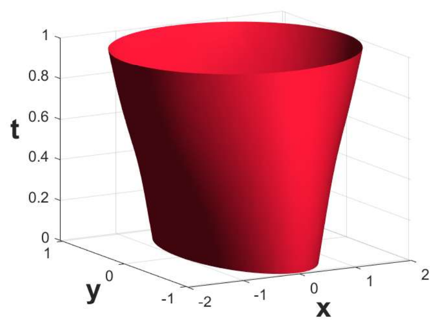
Input data sites ($\boldsymbol{x}_i$) include points on the surface to be reconstructed and normal to the points both inside and outside of the surface. An example in 2D is shown in Fig. 1. The data values used at these points are $s(\boldsymbol{x}_i) = 0$ for points that lie on the surface, $s(\boldsymbol{x}_i) = -1$ inside the surface and $s(\boldsymbol{x}_i) = 1$ outside the surface. The additional points that do not lie on the surface to be reconstructed are introduced to create a gradient in the function, such that a surface rather than a volume is created. Moreover, the additional points are to avoid the trivial solution of the function being zero in the whole domain ($f(\boldsymbol{x}) = 0$).

The linear system in Eqn. (3) is subsequently solved using the aforementioned input data sites($s(\boldsymbol{x}_i)$) and values. Eqn. (2) is then used to form the univariate function and can be used to evaluate the function values at different points in the domain. Points that have a value of zero ($s(\boldsymbol{x}) = 0$) lie on the surface which can be reconstructed.

Multivariate interpolation is additionally achieved through the use of RBFs to create a morphing sequence between two surfaces. An example of 2D morphing between a circle and an ellipse can be seen in Fig. 1. Morphing of a surface is achieved using the same method with an additional time coordinate. The first surface is placed at $t = t_1$ and the final surface at $t = t_N$. The additional coordinate increases the spatial dimensions of the input data sites ($\boldsymbol{x}_i \in R^3$). Radial basis functions allow for interpolation in multi-dimensional spaces without an increase in complexity. The implicit function can be evaluated at different time steps to form the morphing sequence (Fig. 1).



(a)



(b)

Figure 1: (a) Function values used to generate the implicit function that describes the whole surface. Points not lying on the surface are equidistant along the unit normal at each point (b) Morphing between a 2D ellipse and a circle.

2.1.2 Local Support Radial Basis Functions

Developments in compactly supported radial basis functions were made to improve interpolation results, where inaccuracies occur due to the high condition number of the densely populated RBF matrices [Wendland (1995)]. Compact support introduces sparse matrices that result in linear systems (Eqn. (3)) which are faster to solve and provide, in specific cases, more accurate results.

The RBF matrix is made sparse through scaling of the basic function used, $\phi(r)$, by setting $r \in [0, \varepsilon]$, the cut off radius being $r = \varepsilon$. This reduces the neighbourhood considered for each point in the domain, $\boldsymbol{x}$, from all data sites, $\boldsymbol{x}_i$ to the nearest, k, neighbours, where it is assumed that $k \ll N$ (the total number of points in the domain). In order to obtain the nearest neighbours, k, at each point, $\boldsymbol{x}_i = \boldsymbol{x}_1, ..., \boldsymbol{x}_N$, a k-dimensional tree algorithm has been used which firstly, rearranges the node information in O(N log N) operations, and secondly a further O(N log N) operations find a specified number of nearest neighbours to all nodes [Fornberg & Flyer (2015)]. For this, the object function, *knnsearch* in MATLAB has been used. Interpolation of scattered data is achieved by solving the same linear system (as in Eqn. (3)) but with a sparse matrix:

$$\begin{pmatrix} \phi(\|\boldsymbol{x}_1-\boldsymbol{x}_1\|) & \phi(\|\boldsymbol{x}_1-\boldsymbol{x}_2\|) & 0 & & \\ \phi(\|\boldsymbol{x}_2-\boldsymbol{x}_1\|) & \phi(\|\boldsymbol{x}_2-\boldsymbol{x}_2\|) & \phi(\|\boldsymbol{x}_2-\boldsymbol{x}_3\|) & \ddots & \\ 0 & \phi(\|\boldsymbol{x}_3-\boldsymbol{x}_2\|) & \ddots & \ddots & \\ & \ddots & \ddots & \ddots & \phi(\|\boldsymbol{x}_{N-1}-\boldsymbol{x}_N\|) \\ & & & \phi(\|\boldsymbol{x}_N-\boldsymbol{x}_{N-1}\|) & \phi(\|\boldsymbol{x}_N-\boldsymbol{x}_N\|) \end{pmatrix} \tag{4}$$

where for this example all data sites lie on a uniform grid and $\boldsymbol{x}_i \in R^1$ for $k = 3$ neighbouring points. The sparsity of the matrix depends on how the points are ordered in the physical domain.

Alternative functions can be used to interpolate such as Wendland's compactly supported functions which have continuity at the edge of the compact support [Wendland (1995)]. However for the purposes of this study, basic MQ functions are used along with local support to develop finite difference weights for the RBF-FD method (Section 2.2.2) and as the basis for the Local RBF Collocation Method (LRBFCM) introduced in Section 2.2.3.

2.2 RBFs for PDEs

In order to solve linear PDEs two methods were used: Kansa's method and RBF Finite Difference (RBF-FD). For non linear PDEs, the Local Radial Basis Function Collocation Method (LRBFCM) has also been used.

2.2.1 Kansa's Method

For Kansa's method the function value at a point $\boldsymbol{x}$ is approximated by using Eqn. (2). This expression can be differentiated to give:

$$\frac{\partial^n u(\boldsymbol{x})}{\partial x^n} = \sum_{i=1}^{N} \lambda_i \frac{\partial^n}{\partial x^n}\{\phi(\|\boldsymbol{x}-\boldsymbol{x}_i\|)\} \tag{5}$$

and therefore this approximation can be applied to solve PDEs. The linear PDE:

$$L\{u(\boldsymbol{x})\} = f(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega \tag{6}$$

with boundary conditions:

$$G\{u(\boldsymbol{x})\} = g(\boldsymbol{x}), \quad \boldsymbol{x} \in \partial\Omega \tag{7}$$

is considered, where L and G represent differential operators and $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$ are functions of $\boldsymbol{x}$, for the domain (Ω) and the boundary ($\partial\Omega$) respectively. The approximation from Eqn. (5) can be applied to points in the domain and equated to $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$ respectively in order to find the coefficients, λ_i [Franke & Schaback (1998)]. This gives:

$$\sum_{i=1}^{N} \lambda_i L\{\phi(\|\boldsymbol{x}_m-\boldsymbol{x}_i\|)\} = f(\boldsymbol{x}_m) \tag{8}$$

and

$$\sum_{i=1}^{N} \lambda_i G\{\phi(\|\boldsymbol{x}_j - \boldsymbol{x}_i\|)\} = g(\boldsymbol{x}_j) \tag{9}$$

where $\boldsymbol{x}_m$ represents a point in the domain and $\boldsymbol{x}_j$ a point on the boundary. Applying this to the whole domain of points gives the matrix equation:

$$\begin{pmatrix} f(\boldsymbol{x}_1) \\ f(\boldsymbol{x}_2) \\ \vdots \\ f(\boldsymbol{x}_m) \\ g(\boldsymbol{x}_{(m+1)}) \\ \vdots \\ g(\boldsymbol{x}_N) \end{pmatrix} = \begin{pmatrix} L(\Phi_{11}) & L(\Phi_{12}) & ... & L(\Phi_{1N}) \\ L(\Phi_{21}) & L(\Phi_{22}) & ... & L(\Phi_{2N}) \\ \vdots & \vdots & \vdots & \vdots \\ L(\Phi_{m1}) & L(\Phi_{m2}) & ... & L(\Phi_{mN}) \\ G(\Phi_{(m+1)1}) & G(\Phi_{(m+1)2}) & ... & G(\Phi_{(m+1)N}) \\ \vdots & \vdots & \vdots & \vdots \\ G(\Phi_{N1}) & G(\Phi_{N2}) & ... & G(\Phi_{NN}) \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_m \\ \lambda_{(m+1)} \\ \vdots \\ \lambda_N \end{pmatrix} \tag{10}$$

for a domain of N total points and m internal points, where Φ_{ij} represents $\phi(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|)$. The linear system can then be solved to find the unknown coefficients, λ_i. The function values for different points can be found by substituting back into Eqn. (2) [Bernal (2010)].

There are some issues with the Kansa method. When using this method with globally supported RBFs, the matrices created are very dense, making them numerically inefficient to work with. Additionally these matrices generally have a high condition number which can affect the accuracy of the solution when solving the matrices of equations. A method based on locally supported RBFs is needed to solve this issue. Errors can also arise when using Kansa's method through taking derivatives of the interpolation matrix. In order to solve this issue an indirect RBF approach could be taken. In this method the RBFs are used to represent the highest order derivative in the problem (for example the second derivative, $\frac{\partial^2 u}{\partial x^2}$) and then it is integrated to find the lower order derivatives, as opposed to the direct method used by Kansa where the RBFs are used to represent the function u and subsequently get differentiated. This has been found to give more accurate results, as integration is a smoothing operation and therefore the approximating functions are smoother [Mai-Duy & Tanner (2005)]. However, this method is very computationally demanding. Use of a local RBF based Finite Difference method would also solve this issue, as well as being less computationally demanding. As this method is based on locally supported RBFs it would also create less dense and ill-conditioned matrices compared to the global Kansa method.

2.2.2 RBF Finite Difference

In the Finite Difference method the differential operator at a point is approximated by a weighted sum of the function values at other points in the domain. For this work a local FD method is employed, with a stencil of k nearest neighbours being used, as discussed in Section 2.1.2. For a PDE as described in Equations (6) and (7) the differential operation L at point $\boldsymbol{x}_i$ can be approximated as:

$$L\{u_i\} = \sum_{j=1}^{k} w_j u_j \tag{11}$$

where u_i is the function value at point i, w are the stencil weights and k is the number of points in the stencil. The RBF approximation, as given in Eqn. (5), can be applied to

$L\{u_i\}$ and u_j. This gives:

$$\begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_k \end{pmatrix} = \begin{pmatrix} \Phi_{11} & \Phi_{12} & ... & \Phi_{1k} \\ \Phi_{21} & \Phi_{22} & ... & \Phi_{2k} \\ \vdots & \vdots & \vdots & \vdots \\ \Phi_{k1} & \Phi_{k2} & ... & \Phi_{kk} \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_k \end{pmatrix} \tag{12}$$

and

$$L\{u_i\} = \begin{pmatrix} L(\Phi_{11}) & L(\Phi_{12}) & ... & L(\Phi_{1k}) \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_k \end{pmatrix} \tag{13}$$

which can be written as

$$u = A\lambda \tag{14}$$

and

$$L\{u_i\} = b\lambda \tag{15}$$

respectively, where A is the $k \times k$ matrix (where k is the number of points in the stencil), b is the $1 \times k$ vector and λ is the $k \times 1$ column vector of weights. From this λ can be eliminated in order to find the stencil weights in terms of A and b:

$$w = bA^{-1}. \tag{16}$$

This process is repeated to find the stencil weights for all points in the domain. From this, a global weight matrix, W, is constructed with the stencil weights in the appropriate places in each row giving the equation:

$$\begin{pmatrix} f(\boldsymbol{x}_1) \\ f(\boldsymbol{x}_2) \\ \vdots \\ f(\boldsymbol{x}_m) \\ g(\boldsymbol{x}_{(m+1)}) \\ \vdots \\ g(\boldsymbol{x}_N) \end{pmatrix} = Wu \tag{17}$$

[Flyer *et al.* (2014)]. The weights for boundary nodes are dealt with using the same method. For Dirichlet boundary conditions a weight of 1 in the appropriate place (to give $u_j = g(\boldsymbol{x}_j)$) can be used. For Neumann boundary conditions the appropriate Finite Difference scheme would need to be used. The system of equations as given in Eqn. (17) can then solved to find function values u. For N points in the domain, W will be a $N \times N$ matrix but will only have k non-zero values in each row, making it a sparse matrix. The appropriate shape factor, $c = \sqrt{5/N}$, is chosen (as discussed in Section 2.1).

2.2.3 Local Radial Basis Function Collocation Method

The Local Radial Basis Function Collocation Method (LRBFCM) was developed by Sarler & Vertnik (2006) to solve the heat diffusion equation. This local method for solving PDEs is based on interpolation (collocation), using RBFs, of overlapping sub-domains. Similar to compact support RBFs and the RBF-FD method, it aims to mitigate densely populated, ill-conditioned matrices resulting from globally supported RBF matrices (Eqn.

(2)). However, both when compactly supported RBFs or the RBF-FD method are used, the linear system solved includes large sparse matrices (Eqn. (4), (17)). When using LRBFCM the interpolation matrix is much smaller in size. The function values are approximated by:

$$u({}_j\boldsymbol{x}) = \sum_{i=1}^{k} {}_j\lambda_i \phi(\|{}_j\boldsymbol{x} - {}_j\boldsymbol{x}_i\|), \qquad {}_j\boldsymbol{x} \in \Omega \notin \partial\Omega \tag{18}$$

where subscript j denotes the sub-domain (local domain of influence) being solved. The central nodes for each sub domain used do not lie on the boundary. The derivatives of the PDE are approximated using Kansa's method (Section 2.2.1) which is applied at multiple smaller systems of k points. Each local domain includes $k = 3$ neighbouring points for a 1D domain and $k = 5$ points for a 2D domain, resulting in a linear interpolation system which is solved independently for every sub-domain:

$${}_ju = {}_jA_j\lambda \tag{19}$$

3 Applications

Different applications have been considered to the solution of different partial differential equations using and the RBF methods described in the section above. Applications include the solution of linear PDEs, surface reconstruction problems and non linear solvers.

3.1 Linear PDEs

The methods described in Section 2.2 were used to solve elliptic, hyperbolic and parabolic partial differential equations, namely the Poisson equation (elliptic), the heat equation (parabolic) and the wave equation (hyperbolic). Initially the Poisson equation:

$$\nabla^2 u(\boldsymbol{x}) = f(\boldsymbol{x}) \qquad x \in \Omega \tag{20}$$

subject to Dirichlet boundary conditions:

$$u(\boldsymbol{x}) = g(\boldsymbol{x}) \qquad x \in \partial\Omega \tag{21}$$

where $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$ are functions of $\boldsymbol{x}$ and Ω and $\partial\Omega$ are the domain and boundaries respectively. This was solved in 2D using both the Kansa and RBF-FD methods (Section 2.2). The heat equation in two dimensions is given as:

$$\frac{\partial u}{\partial t} = \alpha \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \tag{22}$$

where α is the diffusivity of the medium. In comparison to the Poisson equation this equation has the additional dimension of time. An explicit Euler scheme could be used for the time approximation, i.e:

$$\frac{\partial u}{\partial t} = \frac{u^{n+1} - u^n}{\Delta t} \tag{23}$$

where u^n is the function value at time-step n and Δt is the length of the time step. This can be used in combination with an RBF based discretisation, with the RBFs being used to approximate the double derivatives of u with respect to x and y using Eqn. (5). This gives:

$$u^{n+1} = u^n + \Delta t \alpha \sum_{i=1}^{N} \lambda_i \nabla^2 \phi(\|\boldsymbol{x} - \boldsymbol{x}_i\|). \tag{24}$$

where $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$. By using RBFs to approximate the function values, u^n, the coefficients, λ can be solved for (ie. solving Eqn. (3)) and then used to approximate the derivatives and find u^{n+1} to update the time step. The process can then be repeated to move forward in time as desired. When coding this the interpolation matrices for u and $\nabla^2 u$ can be set up at the start of the code and a simple 'for' loop used to solve for the coefficients and update the time-step.

An implicit method was also tested, with the dimension of time being treated in the same way as the spatial dimensions (ie. the 3^{rd} coordinate). Kansa's method or the RBF-FD was then applied in the same way.

Following the same principle, the wave equation can be solved using an explicit Euler scheme and both the Kansa and RBF-FD methods. The wave equation in two dimensions is given as:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \tag{25}$$

where c is the wave speed. The second time derivative can be approximated as follows:

$$\frac{\partial^2 u}{\partial t^2} = \frac{u^{n+1} - 2u^n + u^{n-1}}{\Delta t^2}, \tag{26}$$

Hence, the wave equation can be expressed as:

$$u^{n+1} = 2u^n - u^{n-1} + c^2 \Delta t^2 \sum_{i=1}^{N} \lambda_i \nabla^2 (\phi(\|\boldsymbol{x} - \boldsymbol{x}_i\|)) \tag{27}$$

$$\sum_{i=1}^{N} \lambda_i \left(c^2 \frac{\partial^2}{\partial x^2} + c^2 \frac{\partial^2}{\partial y^2} - \frac{\partial^2}{\partial t^2} \right) (\phi(\|\boldsymbol{x} - \boldsymbol{x}_i\|)) = 0 \tag{28}$$

The solution process is as with solving the heat equation. An issue arises when computing the solution at time step $n = 1$, as the stencil includes the term u^{n-1}. However, using the initial condition: $u^0 = 0$, it can be found that at this time step, $u^{-1} = u^1$.

3.2 Function Reconstruction

An edge segmentation technique is firstly introduced as preparation for the surface reconstruction dataset. Two surface reconstruction methods, Poisson Surface Reconstruction and Hermite RBF implicits, are explained in greater detail. Images are often noisy and structurally complex. Extracting the key points to reconstruct the image can be helpful to get the general layout of the image thus only essential information remains. To do so, Canny edge detection is employed to remove the noise and thinning the edges to 1 pixel width in the image [Canny (1986)]. The multi-stage algorithm implemented for a 2D image is as follows:

1. Gaussian filter: Remove general background noise.

$$Ga(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{29}$$

2. Sobel operator: Obtain the gradient changes and orientations in the x and y directions and sum up the two.

3.3 Methods for least squares minimisation for non linear problems

Non linear least squares methods can be used to approximate a solution to non linear PDEs. The least squares method consists of approximating a solution to an overdetermined system by minimising the square deviations between the observed results and the fitted value of the model. Non linear least squares approximate the solution from a starting point x_o which converges to a minimiser x^* by successive iterations, refining parameters which enforce a descending condition. It minimises a continuous function $||f(x)||$, for which the gradient f' can be computed, to find:

$$x^* = argmin_x\{F(x)\} \tag{30}$$

where:

$$F(x) = \frac{1}{2}\sum_{i=1}^{m}(f_i(x))^2 = \frac{1}{2}||f(x)||^2 = \frac{1}{2}f(x)^T f(x) \tag{31}$$

where m is the number of given functions. Non linear least squares methods aim to find a local minimiser (x^*) for F, an argument vector that gives a minimum value of F inside a region, and thus enforces the descending condition:

$$F(x^*) \leq F(x) \qquad for \qquad ||x - x^*|| < \delta \tag{32}$$

where δ is a small, positive number. For the variation of a function F in the direction h, its Taylor expansion is given by:

$$F(x+h) = F(x) + h^T g + \frac{1}{2}h^T H h + O(||h||^3) \tag{33}$$

where g is the gradient $F'(x)$ and H is the Hessian ($H \equiv F''(x)$). If $||h||$ is sufficiently small, the sufficient condition for a local minimiser is to assume that x^* is a stationary point and that $F''(x)$ is positive definite. For each iteration towards the local minimiser, the current error e_k is defined as:

$$e_k = x_k - x^* \tag{34}$$

where k denotes the current iteration. The aim of iterative descent methods is to achieve fast convergence, where we distinguish between linear, Eqn. (35), and quadratic convergence, Eqn. (36):

$$||e_{k+1}|| \leq a||e_k|| \tag{35}$$

$$||e_{k+1}|| = O(||e_k||^2) \tag{36}$$

In order to satisfy the descending condition, as presented in Eqn. (32), and find a local minimiser (Eqn. (30)) in each step of the iteration, descent methods consist of, firstly, finding a descent direction h_d, and secondly, finding an appropriate step length $F(x + \alpha h)$ that gives an appropiate decrease in the F-value. The descent direction h is defined for F at x if $h^T F'(x) \leq 0$ [Madsen *et al.* (2004*b*)]. For this definition, different descending methods have been used for the solution of non linear PDEs. The Burgers' equation, in further sections, has been solved using both the Gauss-Newton method and the Levenberg-Marquadt method.

3.3.1 The Gauss-Newton Method

The Gauss-Newton (G-N) method is based on a linear approximation to the components of cost function **f**, L(h). Inserting the Taylor expansion from Eqn. (33) in the definition of the descending condition, Eqn. (32), it can be expressed as:

$$F(x+h) \simeq L(h) = \frac{1}{2}l(h)^T l(h) = \frac{1}{2}f^T f + h^T J^T f + \frac{1}{2}h^T J^T J h \tag{37}$$

where $f = f(x)$ and $J = J_{ij}(x) = \frac{\partial f_i}{\partial x_j}$ is the Jacobian matrix. The Gauss-Newton step, h_{gn} minimises L(h), and the gradient and the Hessian of L are defined as:

$$L'(h) = J^T f + J^T J h \qquad L''(h) = J^T J \tag{38}$$

where $L'(0) = F'(x)$ and the second derivative $L''(h)$ is symmetric and independent of h. If the Jacobian has full rank, i.e. linearly independent columns, $L''(h)$ is positive definite, hence $L(h)$ has a unique minimiser found by solving the following equation:

$$(J^T J)h_{gn} = -J^T f \tag{39}$$

$$x = x + \alpha h_{gn}$$

where α is found by line search, as described in Madsen *et al.* (2004*a*). However, for this study it has been considered that $\alpha = 1$ in all steps, as in the classical Gauss-Newton method. The method has linear convergence at initial iterations, although quadratic convergence is achieved at final stages of the iteration. The Gauss-Newton algorithm is the following:

(a) Set an initial guess for the solution $x = x_o$.

(b) Compute the Gauss Newton descent direction, $(J^T J)h_{gn} = (-J^T f)$, for a given function, f and its Jacobian, $J(x)$.

(c) Update the solution by stepping in the required direction $x_{new} = x + h_{gn}$.

(d) Repeat until $||f(x)||_\infty \leq \epsilon_1$ or $k = k_{max}$.

In this work, the G-N algorithm is repeated until the function reaches $\epsilon_1 = 10^{-5}$, or it reaches a maximum number of iterations $k_{max} = 200$, to prevent it from an infinite loop.

3.3.2 The Levenberg-Marquadt Method

The Levenberg-Marquadt (L-M) method is a hybrid descent method that suggests a modification to the Gauss-Newton method by including a damping parameter μ. Thus, the function minimised becomes $L(h) + \frac{1}{2}\mu h^T h$, and the step h_{lm} is defined as:

$$(J^T J + \mu I)h_{lm} = -J^T f \qquad \textit{and } \mu \geq 0 \tag{40}$$

where $f = f(x)$ and $J = J_{ij}(x) = \frac{\partial f_i}{\partial x_j}$, and I is the identity matrix. The damping parameter constrains large steps and it has several effects: firstly, for positive values of μ, the minimised matrix $L_{lm}(h)$ is positive definite which ensures the step h_{lm} is in a descent direction. Secondly, for very large values of μ, the step is approximated as:

$$h_{lm} = -\frac{1}{\mu}F'(x) \tag{41}$$

which is, essentially, a step in the steepest descent direction, a method where the descending condition is given by $h_{sd} = F'(x)$. The method is particularly useful if the current iteration is far from the solution, but its convergence is linear and often slow Madsen *et al.* (2004*a*). Thirdly, for small values of μ, $h_{lm} \simeq h_{gn}$, which is desirable at final stages of the iteration when x is close to x^*, so quadratic convergence can be achieved. The Levenberg-Marquadt method is a hybrid method combining the steepest descent and the Gauss-Newton method. It does not require a specific line search as the damping parameter influences the size of the step. The choice of the initial damping value is related to the size of elements in $D_0 = J(x_0)^T J(x_0)$, and is defined by $\mu_o = \tau max_i\{D_{ii}\}$, where τ is chosen by the user [Madsen *et al.* (2004*b*)]. In this research, a value of $\tau = 10^{-3}$ is shown to be a good approximation. The damping value changes through iterations and the updating is controlled by the *gain ratio*:

$$\varrho = \frac{F(x) - F(x + h_{lm})}{L(0) - L(h_{lm})} \tag{42}$$

where the denominator is the gain predicted by the linear model and it is defined as:

$$L(0) - L(h_{lm}) = \frac{1}{2}h_{lm}^T(\mu h_{lm} - g) \tag{43}$$

where $g = J^T f$. Both terms are positive, hence the denominator of the gain ratio is positive. A large value of ϱ indicates that $L(h_{lm})$ is a good approximation to the function, so μ decreases, and the step gets closer to a G-N step. However, if ϱ is small or negative it means that $L(h_{lm})$ is a poor approximation, so the damping value increases in the next step aiming to get closer to a steepest descent direction [Madsen *et al.* (2004*b*)]. Regarding the stopping criteria, the iteration is stopped when the function $(J^T f)$ is sufficiently minimised, i.e. when it reaches a value of $\epsilon_1 = 10^{-5}$, or when the change of x becomes sufficiently small,

$$||x_{new} - x|| \leq \epsilon_2(||x|| + \epsilon_2) \tag{44}$$

where ϵ_2 is set to 10^{-5}. Thus, the Levenberg-Marquadt algorithm is the following:

(a) Set an initial guess for the solution.

(b) Compute the Levenberg-Marquadt descent direction, $(J^T J+\mu I)h_{lm} = (-J^T f)$.

(c) Stop iteration if the change in x is small, i.e. stopping condition described in Eqn. (44).

(d) Update the solution by stepping in the required direction $x_{new} = x + h_{lm}$.

(e) If ϱ is > 0, accept the new step and set $\mu = \mu * max(\frac{1}{3}, 1 - (2\varrho - 1)^3)$.
If ϱ is < 0, compute $\mu = \mu\nu$ and $\nu = 2\nu$.

(f) Repeat until $||g(x)||_\infty \leq \epsilon_1$ or $k = k_{max}$.

where ν is a user-defined constant. In this work, the constant has been set to $\nu = 2$.

3.4 Non linear PDEs: example using Burgers' Equation

RBFs are proven to be a valuable tool for the discretisation of PDEs and least square methods can be used to approximate a solution to non linear problems. Therefore, the 1D and 2D Burgers equation [Bateman (1915)] is numerically solved as a classical test case:

$$\begin{aligned}\frac{\partial u_x}{\partial t} + u_x\frac{\partial u_x}{\partial x} + u_y\frac{\partial u_x}{\partial y} &= \frac{1}{Re}\left[\frac{\partial^2 u_x}{\partial x^2} + \frac{\partial^2 u_x}{\partial y^2}\right]\\ \frac{\partial u_y}{\partial t} + u_x\frac{\partial u_y}{\partial x} + u_y\frac{\partial u_y}{\partial y} &= \frac{1}{Re}\left[\frac{\partial^2 u_y}{\partial x^2} + \frac{\partial^2 u_y}{\partial y^2}\right]\end{aligned} \tag{45}$$

where Re is the Reynolds number of the flow. The Burgers' equation is a conservation of momentum equation with a dissipative term arising due to fluid viscosity. The schemes provided in this section are differentiated by different time and spatial discretisations. The spatial discretisations have been conducted using the Global Kansa, LRBFCM and RBF-FD methods, and time derivatives were discretised using both an explicit Euler scheme, which has been previously used by Sarler *et al.* (2012), and an implicit Euler scheme, which is a novel approach conducted in this work. The implicit scheme is used to ensure stability at $Re > 100$. As Sarler *et al.* (2012) notes, for high Reynolds numbers, the flow becomes convective and oscillations arise which increase rapidly, which Sarler *et al.* (2012) removed through upwinding.

The initial conditions and Dirichlet boundary conditions used are taken from the analytical solution of the Burgers' equation. For the 1D Burgers' equation the analytical solution is given by Malfliet (1992), computed using the hyperbolic tangent method:

$$u(x,t) = \alpha + \frac{2}{Re} tanh(x-\alpha) \tag{46}$$

where α is a free parameter that arises from the analytical solution and bounds the solution to a specific domain. The parameter is set to $\alpha = 0.1$ as in Khater *et al.* (2008) and Ali *et al.* (2011). The coupled 2D Burgers' equation is solved analytically by Fletcher (1983) using the Hopf-Cole transformation. The analytical solution is given by:

$$\begin{aligned}u_x(x,y,t) &= \frac{3}{4} - \frac{1}{4[1+exp((-4x+4y-t)(\frac{Re}{32}))]}\\ u_y(x,y,t) &= \frac{3}{4} + \frac{1}{4[1+exp((-4x+4y-t)(\frac{Re}{32}))]}\end{aligned} \tag{47}$$

3.4.1 Explicit Time Scheme and Global RBF

The velocity, u, is approximated in the whole domain using global RBF interpolation (Section 2.1.1):

$$u(x,t) = \sum_{i=1}^{N} \lambda_i \phi(\|x - x_i\|), \qquad x \in \Omega \tag{48}$$

where $i = 1, ..., N$ are the indices for all the points in the domain (Ω) that are used for the interpolation. The derivatives in space are approximated using Kansa's method (Section 2.2.1) and the time derivative of the Burgers' equation is approximated using an Euler upwind scheme (explicit). The discretised Burgers' equation is given by:

$$\frac{u^{n+1} - u^n}{\Delta t} + u^n \sum_{i=1}^{N} \lambda_i \frac{\partial}{\partial x}\phi(\|x - x_i\|) = \frac{1}{Re}\sum_{i=1}^{N} \lambda_i \frac{\partial^2}{\partial x^2}\phi(\|x - x_i\|), \qquad x \in \Omega \notin \partial\Omega \tag{49}$$

where Δt is the time step size, n is the time step index and the equation is solved for the interior nodes of the domain only. The update formula for the velocity in the inner domain is as follows:

$$u^{n+1} = u^n - \Delta t u^n B\lambda + \frac{\Delta t}{Re} C\lambda \tag{50}$$

where B and C are the RBF matrices formed using Kansa's method. Starting with the initial condition, given by Eqn. (46), the velocity is updated at every time step giving the required solution for the inner domain ($u^{n+1} = u(x, t + \Delta t)$). The boundary conditions imposed are given by Eqn. (46) and the coefficients λ for the next time step are recalculated using the interpolated function (Eqn. (48)). The ill-conditioned linear system is solved iteratively using the `gmres` solver available in MATLAB.

3.4.2 Explicit Time Scheme and Local RBF

The Burgers' equation has also been solved using the local RBF collocation scheme developed by Sarler & Vertnik (2006). The velocity field in the inner domain (${}_j\boldsymbol{x} \in \Omega \notin \partial\Omega$) is approximated using the LRBFCM method (Eqn. (18)). Additionally, the time derivative is discretised using an Euler upwind scheme (explicit) and the spatial derivatives using a local Kansa method as described in Section 2.2.3. The velocity is updated at every time step using the following formula:

$${}_ju^{n+1} = {}_ju^n - \Delta t_j u^n{}_j B_j \lambda + \frac{\Delta t}{Re}{}_j C_j \lambda \tag{51}$$

where Δt is the time step and the ${}_jB$, ${}_jC$ matrices are formulated using the Kansa method as in Eqn. (50). The velocity is calculated at the central node of every overlapping sub-system for ever time iteration. The boundary conditions are imposed following a time step. Points in each subsystem ${}_jx_i$ are scanned, and if they lie on the boundary their velocity value is imposed using Eqn. (46). The local matrices are smaller ($k \ll N$), hence the system is solved using lower-upper decomposition (LU) through the backslash command in MATLAB [Mathworks (2020)].

3.4.3 Explicit Time Scheme and RBF-FD

The Burgers' equation has also been solved using the a local RBF Finite Difference (RBF-FD) method. The discretisation weights for the spatial derivatives are calculated as explained in Section 2.2.2, where the differential operators are approximated by a weighted sum of RBFs from the neighbouring points around it, obtained using a *k-d* tree algorithm. Specifically, using an object function *knnsearch* from MATLAB's statistics tool box. With N nodes accross the full domain, the RBF-FD stencil is centered at each of these, and is extended over a total k nearest neighbours, where it is assumed that k $\ll$ N. Thus, the weights are solved by lower-upper(LU) decomposition, through the backslash command in MATLAB. The time derivative is discretised using an Euler upwind scheme (explicit) and the final discretised solution of the 1D Burgers' equation is given by:

$$u^{n+1} = u^n - \Delta t(u^n(W_x u^n_\Omega)) + \frac{\Delta t}{Re}(W_{xx} u^n_\Omega) \tag{52}$$

where u is the velocity vector of the interior nodes of the domain (excluding the boundary), u_Ω is the velocity vector of all the points in the domain and W_x and W_{xx} are the global weight matrices for the first and second spatial derivatives respectively, where the number of rows corresponds to the number of interior nodes of the velocity vector. The inner domain velocity is updated at each time step, and the initial condition and the boundary values are imposed, with u_Ω being updated from the analytical solution (Eqn. (46)).

The 2D Burgers' equation (Eqn. (47)) has been similarly discretised and solved for the y dimension in all three explicit schemes. This can be easily achieved since no coupling is required (Eqn. (50), (51), (52)), as the solver uses previous time step information to move forward in time.

3.4.4 Implicit Time Scheme and Global RBF

A global RBF approximation to the velocity (Eqn. (48)) and an Euler downwind (implicit) time scheme is used along with Kansa's method (Section 2.2.1) for the spatial derivatives. The discretisation is as follows:

$$\frac{u^{n+1} - u^n}{\Delta t} + u^{n+1}\frac{\partial u}{\partial x}\Big|_{t=\Delta t(n+1)} = \frac{1}{Re}\frac{\partial^2 u}{\partial x^2}\Big|_{t=\Delta t(n+1)} \tag{53}$$

$$\begin{aligned}\sum_{i=1}^{N}\lambda_i\phi(\|x-x_i\|) - u^n + \Delta t\sum_{i=1}^{N}\lambda_i\phi(\|x-x_i\|)\sum_{i=1}^{N}\lambda_i\frac{\partial}{\partial x}\phi(\|x-x_i\|) \\ = \frac{\Delta t}{Re}\sum_{i=1}^{N}\lambda_i\frac{\partial^2}{\partial x^2}\phi(\|x-x_i\|), \quad x \in \Omega \notin \partial\Omega\end{aligned} \tag{54}$$

where the velocity is only calculated at the interior nodes of the domain. This leads to a non linear system of equations to be solved iteratively:

$$f(\lambda) = A\lambda + \Delta t(A\lambda)\circ(B\lambda) - \frac{\Delta t}{Re}(C\lambda) - u^n \tag{55}$$

where $\circ$ is the Hadamard product, i.e. an element-wise multiplication. Additionally, A is the RBF matrix used to approximate the velocity, and B, C are the radial basis function matrices discretising the first and second spatial derivatives respectively. The solution is approximated iteratively as discussed in Section 3.3, where solvers require the computation of the Jacobian matrix, which is given by:

$$J(\lambda) = A + \Delta t(A\circ(B\lambda\overrightarrow{\mathbf{1}}^T) + B\circ(A\lambda\overrightarrow{\mathbf{1}}^T)) - \frac{\Delta t}{Re}C \tag{56}$$

where $\overrightarrow{\mathbf{1}} = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \end{pmatrix}^T$. The non linear least squares method used is the Gauss-Newton method, which is also used in Bahadır (2003). Thus, the iterative algorithm is as follows:

(a) Start with an initial guess $u^n(x,t) = \sum_{i=1}^{N} \lambda_i \phi(\|x - x_i\|), \;\; x \in \Omega, \, \lambda = A^{-1}u^n$.

(b) Use λ and $u^n(x,t), \;\; x \in \Omega \notin \partial\Omega$ in $f(\lambda)$ and $J(\lambda)$.

(c) Apply the Gauss Newton method (Section 3.3) to find λ_{new} .

(d) Use λ_{new} to calculate $u^{n+1} = A\lambda_{new}$.

(e) Impose boundary conditions to $u^{n+1}(x,t), \;\; x \in \Omega$.

(f) Move to step 1 with a different initial guess $u^n(x,t) = u^{n+1}(x,t)$ until $t = t_{end}$.

The 2D equation, Eqn. (47), requires a coupling of the system so that the Gauss-Newton method can iteratively solve for the velocity in both directions at the same time. The coupled system of equations produces the following function to be minimised, along with its Jacobian:

$$\begin{aligned} f(\boldsymbol{\lambda}) = &\begin{pmatrix} A & 0 \\ 0 & A \end{pmatrix}\begin{pmatrix} \lambda_x \\ \lambda_y \end{pmatrix} + \Delta t \left[\begin{pmatrix} A & 0 \\ A & 0 \end{pmatrix}\begin{pmatrix} \lambda_x \\ \lambda_y \end{pmatrix}\right] \circ \left[\begin{pmatrix} B & 0 \\ 0 & B \end{pmatrix}\begin{pmatrix} \lambda_x \\ \lambda_y \end{pmatrix}\right] \\ &+\Delta t \left[\begin{pmatrix} 0 & A \\ 0 & A \end{pmatrix}\begin{pmatrix} \lambda_x \\ \lambda_y \end{pmatrix}\right] \circ \left[\begin{pmatrix} C & 0 \\ 0 & C \end{pmatrix}\begin{pmatrix} \lambda_x \\ \lambda_y \end{pmatrix}\right] - \frac{\Delta t}{Re}\begin{pmatrix} D & 0 \\ 0 & D \end{pmatrix}\begin{pmatrix} \lambda_x \\ \lambda_y \end{pmatrix} \end{aligned} \tag{57}$$

$$\begin{aligned} J(\boldsymbol{\lambda}) = &\begin{pmatrix} A & 0 \\ 0 & A \end{pmatrix} + \Delta t \Bigg\{ \begin{pmatrix} A & 0 \\ 0 & A \end{pmatrix} \circ \left[\begin{pmatrix} B & 0 \\ 0 & C \end{pmatrix}\begin{pmatrix} \lambda_x \\ \lambda_y \end{pmatrix}\begin{pmatrix} \overrightarrow{\mathbf{1}} \\ \overrightarrow{\mathbf{1}} \end{pmatrix}^T\right] \\ &+ \begin{pmatrix} B & 0 \\ 0 & C \end{pmatrix} \circ \left[\begin{pmatrix} A & 0 \\ 0 & A \end{pmatrix}\begin{pmatrix} \lambda_x \\ \lambda_y \end{pmatrix}\begin{pmatrix} \overrightarrow{\mathbf{1}} \\ \overrightarrow{\mathbf{1}} \end{pmatrix}^T\right] \Bigg\} \\ &+\Delta t \Bigg\{ \begin{pmatrix} C & 0 \\ 0 & B \end{pmatrix} \circ \left[\begin{pmatrix} 0 & A \\ A & 0 \end{pmatrix}\begin{pmatrix} \lambda_x \\ \lambda_y \end{pmatrix}\begin{pmatrix} \overrightarrow{\mathbf{1}} \\ \overrightarrow{\mathbf{1}} \end{pmatrix}^T\right] \Bigg\} - \frac{\Delta t}{Re}\begin{pmatrix} D & 0 \\ 0 & D \end{pmatrix} \end{aligned} \tag{58}$$

where matrix A is the RBF matrix used to approximate both u_x, u_y through Global RBF interpolation (Section 2.1.1), matrices B, C are the RBF matrices used to discretise $\frac{\partial}{\partial x}, \frac{\partial}{\partial y}$ using Kansa's method respectively, and matrix D corresponds to the discretised laplacian operator $(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2})$. All matrices exclude point centres on boundary points (i.e. $rows < N$), as boundary velocity values are imposed at each time iteration.

3.4.5 Implicit Time Scheme and RBF-FD

The RBF-FD method has also been used to discretise the spatial derivatives combined with an Euler downwind (implicit) time scheme. The discretisation is as follows:

$$\frac{u^{n+1} - u^n}{\Delta t} + u^{n+1}\frac{\partial u}{\partial x}\Big|_{t=\Delta t(n+1)} = \frac{1}{Re}\frac{\partial^2 u}{\partial x^2}\Big|_{t=\Delta t(n+1)} \tag{59}$$

$$u^{n+1} - u^n + \Delta t(u^{n+1}(W_x u_\Omega^{n+1})) = \frac{\Delta t}{Re}(W_{xx}u_\Omega^{n+1}) \tag{60}$$

where u is the velocity vector of the inner domain points (excluding the boundary), u_Ω is the velocity vector of all the points in the domain and W_x, W_{xx} are the global weight matrices for the first spatial derivative and the second spatial derivative respectively with rows corresponding to the inner domain velocity points. The system of equations to be solved using non linear least squares algorithms detailed in Section 3.3 is:

$$f(u^{n+1}) = u^{n+1} - u^n + \Delta t u^{n+1} \circ (W_x u_\Omega^{n+1}) - \frac{\Delta t}{Re}(W_{xx} u_\Omega^{n+1}) \tag{61}$$

$$J(u^{n+1}) = I + \Delta t((W_x u_\Omega^{n+1}) \overrightarrow{\mathbf{1}}^T \circ I) + (W_x \circ (u^{n+1} \overrightarrow{\mathbf{1}}^T)) - \frac{\Delta t}{Re} W_{xx} \tag{62}$$

where $\circ$ is the Hadamard product, an element wise multiplication, and I is the identity matrix. The system aims to minimise the velocity vector in the next time step, u^{n+1}. It has been solved using both the Gauss-Newton and Levenberg-Marquadt least squares methods, using the algorithms detailed in Section 3.3, and both methods are compared in Section 4.5.1. The boundary conditions are imposed at each iteration to get the velocity in the whole domain u^{n+1}. The iterative algorithm is the following:

(a) Use the initial conditions to set a value for u^n, u_Ω^n and an initial guess for $u^{n+1} = u^n$.

(b) Use the weights and velocity components in $f(u^{n+1})$ and $J(u^{n+1})$.

(c) Apply non linear least squares methods (Section 3.3) to find the velocity vector, u^{n+1} .

(d) Update $u^n = u^{n+1}$, and the boundaries at u_Ω^{n+1} to match the next time step.

(e) Repeat from step 2 until $t = t_{end}$.

In order to solve the 2D Burgers' equation, the velocity vector is solved for both dimensions and the following function is minimised using the methods given in Section 3.3:

$$\begin{aligned} f(\boldsymbol{u^{n+1}}) = \begin{pmatrix} u_x^{n+1} \\ u_y^{n+1} \end{pmatrix} &+ \Delta t \left[\begin{pmatrix} I & 0 \\ I & 0 \end{pmatrix} \begin{pmatrix} u_{\Omega_x}^{n+1} \\ u_{\Omega_y}^{n+1} \end{pmatrix} \right] \circ \left[\begin{pmatrix} W_x & 0 \\ 0 & W_x \end{pmatrix} \begin{pmatrix} u_{\Omega_x}^{n+1} \\ u_{\Omega_y}^{n+1} \end{pmatrix} \right] \\ &+ \Delta t \left[\begin{pmatrix} 0 & I \\ 0 & I \end{pmatrix} \begin{pmatrix} u_{\Omega_x}^{n+1} \\ u_{\Omega_y}^{n+1} \end{pmatrix} \right] \circ \left[\begin{pmatrix} W_y & 0 \\ 0 & W_y \end{pmatrix} \begin{pmatrix} u_{\Omega_x}^{n+1} \\ u_{\Omega_y}^{n+1} \end{pmatrix} \right] \\ &- \frac{\Delta t}{Re} \left[\begin{pmatrix} W_L & 0 \\ 0 & W_L \end{pmatrix} \begin{pmatrix} u_{\Omega_x}^{n+1} \\ u_{\Omega_y}^{n+1} \end{pmatrix} \right] - \begin{pmatrix} u_x^n \\ u_y^n \end{pmatrix} \end{aligned} \tag{63}$$

$$\begin{aligned} J(\boldsymbol{u^{n+1}}) = \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix} &+ \Delta t \Bigg\{ \left[\begin{pmatrix} W_x & 0 \\ 0 & W_y \end{pmatrix} \begin{pmatrix} u_{\Omega_x}^{n+1} \\ u_{\Omega_y}^{n+1} \end{pmatrix} \begin{pmatrix} \overrightarrow{\mathbf{1}} \\ \overrightarrow{\mathbf{1}} \end{pmatrix}^T \right] \circ \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix} \\ &+ \left[\begin{pmatrix} W_x & 0 \\ 0 & W_y \end{pmatrix} \circ \left[\begin{pmatrix} u_{\Omega_x}^{n+1} \\ u_{\Omega_y}^{n+1} \end{pmatrix} \begin{pmatrix} \overrightarrow{\mathbf{1}} \\ \overrightarrow{\mathbf{1}} \end{pmatrix}^T \right] \right] \Bigg\} \\ &+ \Delta t \Bigg\{ \left[\begin{pmatrix} 0 & I \\ I & 0 \end{pmatrix} \begin{pmatrix} u_{\Omega_x}^{n+1} \\ u_{\Omega_y}^{n+1} \end{pmatrix} \begin{pmatrix} \overrightarrow{\mathbf{1}} \\ \overrightarrow{\mathbf{1}} \end{pmatrix}^T \right] \circ \begin{pmatrix} W_y & 0 \\ 0 & W_x \end{pmatrix} \Bigg\} - \frac{\Delta t}{Re} \begin{pmatrix} W_L & 0 \\ 0 & W_L \end{pmatrix} \end{aligned} \tag{64}$$

where the matrices W_x, W_y correspond to the global weight matrices used to discretise the spatial derivatives with respect to the x and y direction respectively. W_L is the global weight matrix used to discretise the Laplacian operator in the 2D Burgers' equation.

4 Results and Discussion

The metrics used to show accuracy of results are both the L_2 and L_∞ norms:

$$\begin{aligned} \varepsilon_1 &= L_2(\boldsymbol{u}) = \|\hat{\boldsymbol{u}} - \boldsymbol{u}\|_2 = \sqrt{(\hat{\boldsymbol{u}}_1 - \boldsymbol{u}_1)^2 + \ldots + (\hat{\boldsymbol{u}}_n - \boldsymbol{u}_n)^2}, \\ \varepsilon_2 &= L_\infty(\boldsymbol{u}) = \|\hat{\boldsymbol{u}} - \boldsymbol{u}\|_\infty = \max|\hat{\boldsymbol{u}} - \boldsymbol{u}| \end{aligned} \tag{65}$$

where $\hat{\boldsymbol{u}}$ is the analytical value of the dependent variable in the domain and $\boldsymbol{u}$ is the variable approximated through the relevant numerical scheme, n are the number of points in the domain. The L_2 norm provides the total error and the L_∞ norm, the maximum absolute error.

4.1 Linear PDEs: the Poisson Equation

Results to the Poisson Equation (Eqn. 20) depend on the interior function, $f(\boldsymbol{x})$ and boundary conditions. In order to assess the accuracy of the solution, an example analytical solution is needed to be found. The case was considered for a $[1 \times 1]$ domain with a function $f(\boldsymbol{x}) = 0$ (Laplace's Equation) and Dirichlet boundary conditions $u(0, y) = u(1, y) = u(x, 0) = 0$ and $u(x, 1) = g_1(x)$, where:

$$g_1(x) = \begin{cases} x & 0 \le x \le \frac{1}{2} \\ 1 - x & \frac{1}{2} \le x \le 1. \end{cases} \tag{66}$$

Using the separation of variables technique, the solution can be found to be:

$$u(x, y) = \sum_{n=1}^{\infty} \frac{4\sin(n\pi/2)}{n^2\pi^2\sinh(n\pi)} \sin(n\pi x)\sinh(n\pi y) \tag{67}$$

[Homer *et al.* (2017)]. This solution can be seen plotted in Fig. 2a. This is a challenging test case as the boundary conditions mean the solution has a cusp, and hence the derivative is not continuous. This can be seen in Fig. 2a as there is a peak in the centre of the boundary.

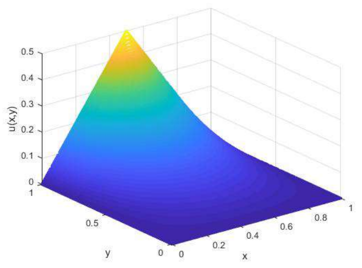
4.1.1 Kansa's Method

Kansa's method was applied to solve the Poisson Equation subject to the boundary conditions given in Section 4.1. In Kansa's method a selection of points is needed to construct the matrix equation as given in Eqn. 10. The weights, λ, were then found through solving this system of equations using the MATLAB solver `mldivide`. These weights could then be used to find the function value at any point in the domain (Eqn. (2)). Initially a regular grid of points (with spacing, Δx) was used to construct the matrix equation, and the weights found were used to calculate the values for a 200×200 grid of points. Fig. 2 shows a plot of the solution obtained from grid spacings of $\Delta x = 0.2$ and $\Delta x = 0.008$, giving a grid of $N = 36$ and $N = 15876$ points respectively. It can be seen that the higher resolution grid gives a better representation of the sharp point of the solution, with the results from the lower resolution grid smoothing out this point. Fig. 3a shows how the error, as quantified by the L_∞ norm, varies with number of grid points. It can be seen that the error generally decreases as the number of points increases. However for the last data point the error increases again. This may be because of the global Kansa method, where the matrix of equations to be solved will get larger as the number of points increases, making it generally less well conditioned. For example, the matrix generated from $N = 36$ grid points has a condition number of 7.121×10^4 whereas for the matrices generated from $N = 10201$ and $N = 15876$ grid points the condition numbers are 3.54×10^9 and 7.02×10^9 respectively. Having an ill-conditioned matrix can affect the accuracy of the MATLAB solver and is likely the reason why there is an increase in error for the final data point. Therefore although having a finer grid resolution will initially increase the accuracy of the solution, the matrices to be solved will become larger and more ill-conditioned as N increases, becoming more computationally demanding to solve and eventually resulting in an increase in error.

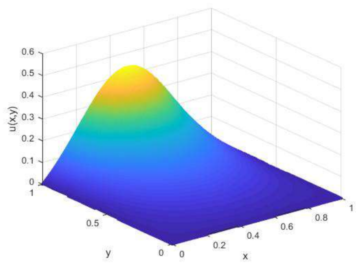
As the RBF approximation is based on the distance between points, not the connectivity, randomly scattered points can also be used. Points in the domain, with x and y coordinate values between 0 and 1, were generated through the `rand` function in MATLAB. Points were also generated on the boundary through enforcing one coordinate value to ensure it sat on a boundary and generating the other coordinate through the `rand` function. These were done in the same proportion of boundary vs. inner points as in the regularly spaced grid, with boundary points evenly split between the 4 edges. The tests were run 5 times for each number of points, with the maximum, minimum and mean L_∞ norm shown in Fig. 3b. The results follow the same pattern as those given by the grid of points, with the average error again decreasing as N increases before increasing for the last data point. The magnitude of results is generally comparable to those given by the evenly spaced grid , however the results given from the grid are more accurate than the average result from the random points. There is a spread of results for each number of points, with the largest error bar being for the last data point. This shows that the accuracy of the results can vary depending on the distribution of points. Overall Kansa's method is shown to be accurate when used with scattered points. However results obtained by an even grid of points vary less and are easier to compare, therefore a even grid of points was used throughout the rest of the report.

4.1.2 RBF-FD

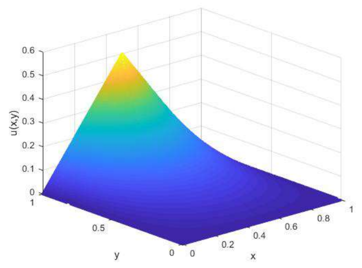
The RBF-FD method was also implemented to solve the Poisson equation. Initially a stencil of the $k = N/6$ closest points was used, where N is the total number of points in the domain. Whereas in the Kansa method once the weights are found the function value for any point in the domain could be calculated, for the RBF-FD method the function values are only found for the points used. This can be seen in



(a) Analytical Solution

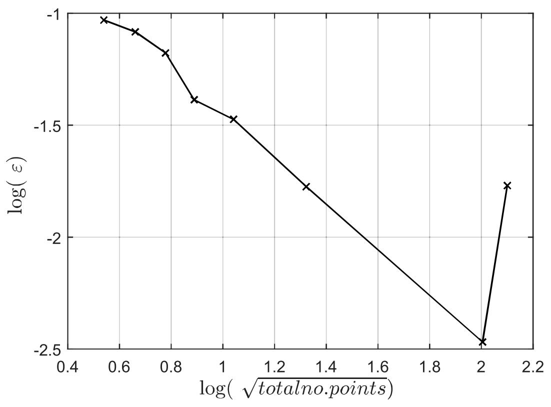


(b) $N = 36$ Points (6×6 Grid).

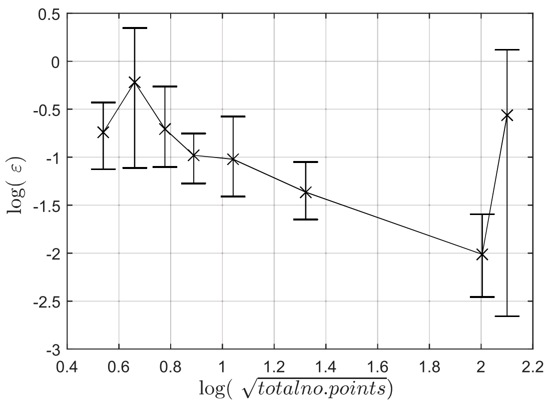


(c) $N = 15876$ Points (126×126 Grid).

Figure 2: Results for the Poisson Equation given by (a) Analytical solution (Eqn. (67)) and (b), (c) grids of different resolutions using Kansa's method.



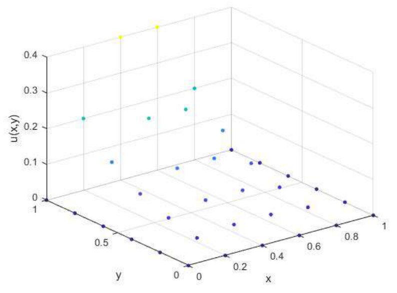
(a)



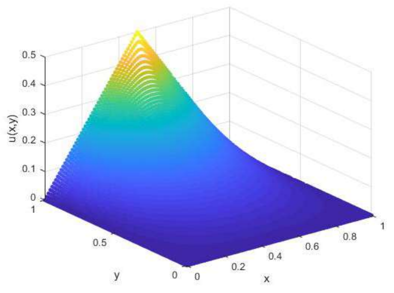
(b)

Figure 3: Graph of $\log(L_\infty)$ vs. $\log(\sqrt{\text{total no. points}})$ for (a) Evenly spaced grid (b) Scattered points using Kansa's method.

Fig. 4. Fig. 5 shows how the L_∞ norm varies with number of points. The error generally decreases as N increases. The error is also smaller than for the results obtained from Kansa's method using the same number of points. One disadvantage of this RBF-FD method is that solving Eqn. 16 for every stencil of points can take a long computational time for a large number of points. In order to resolve this a smaller number of points in the stencil can be used. Results for $k = N/50$ were obtained for comparison, which can also be seen in Fig. 5. The smaller stencil size took less computational time and therefore could be implemented on high resolution grids, however was less accurate than the larger stencil, especially for low resolution grids. Overall this method can give accurate results and the issues of ill-conditioned matrices can be avoided provided an appropriate stencil size for the resolution is chosen.



(a) $N = 36$ Points (6×6 Grid).



(b) $N = 10201$ Points (101×101 Grid).

Figure 4: Results for the Poisson Equation given by grids of different resolutions using RBF-FD (Stencil Size $k = N/6$)

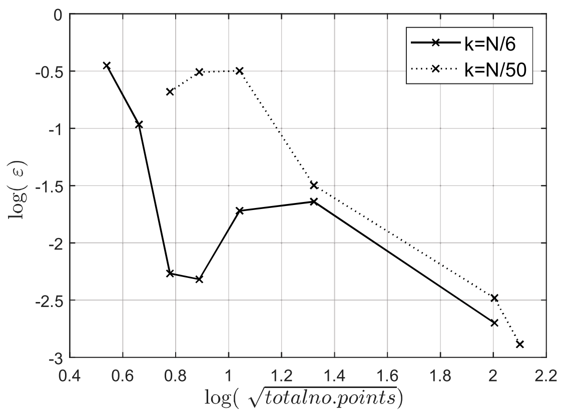


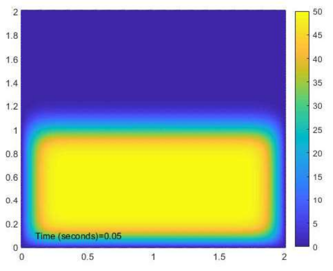
Figure 5: Graph of $\log(L_\infty)$ vs. $\log(\sqrt{\text{total no. points}})$ for the Poisson Equation using RBF-FD.

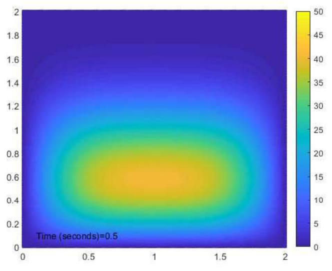
4.2 Linear PDEs: the Heat Equation

In order to assess the accuracy of the solutions obtained from the RBF approximation methods the results were compared to an analytical solution. A solution was found for the situation where a 2D plate of size $2 \times 2m$ with $\alpha = \frac{1}{9}m^2/s$ is initially heated to a temperature of $50°C$ for $y \leq 1$ and $0°C$ for $y > 1$. The plate has Dirichlet boundary conditions, with the edges held at $0°C$. It can be shown that the analytical solution for this situation at position (x, y) and time t is:

$$u(x,y,t) = \frac{200}{\pi^2}\sum_{m=1}^{\infty}\sum_{n=1}^{\infty}(\frac{(1+(-1)^{m+1})(1-\cos(\frac{n\pi}{2}))}{mn}\sin(\frac{m\pi}{2}x)\times\sin(\frac{n\pi}{2}y)e^{-\pi^2(m^2+n^2)t/36} \tag{68}$$

[Daileda (2012)]. This was used to plot the solution for the $2 \times 2m$ plate over a time period of $t_{end} = 1s$, as shown in Fig. 6.


(a) t=0.05s


(b) t=0.5s

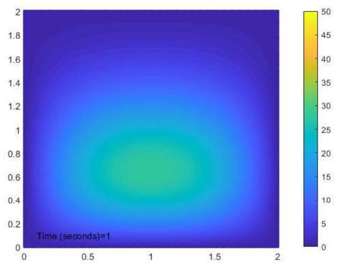

(c) t=1s

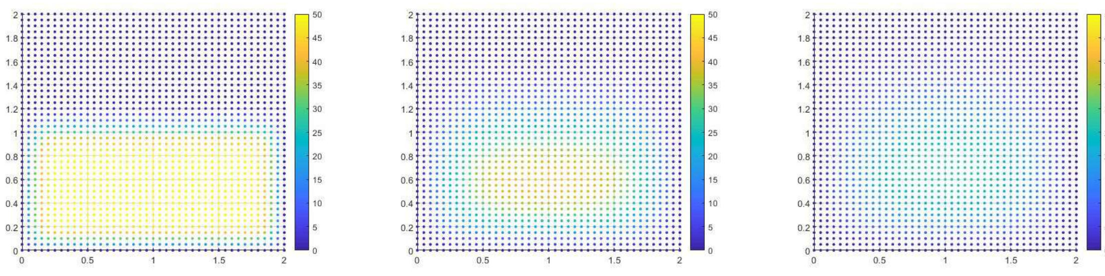
Figure 6: Analytical solution (Eqn. 68.) to the Heat Equation subject to given boundary and initial conditions at 3 time steps.

4.2.1 Explicit Time Scheme

The explicit time scheme, as outlined in Section 3.1, was used to solve the heat equation subject to initial and boundary conditions as given above. This was again ran for a time period of $1s$. Grids of spacing $\Delta x = 0.1m$ and $\Delta x = 0.05m$ were used with different time steps, with the result at $t_{end} = 1s$ being compared to the analytical solution. An example of the results obtained are shown in Fig. 7. It can be seen that these visually appear to follow the analytical solution as given in Fig. 6. Fig. 8 shows how the error, represented using the L_∞ norm, varies with time step size. For large values of Δt (i.e. small values of $t_{end}/\Delta t$) the method will not be stable, with the L_∞ norm being in the order of 10^{20}-10^{30}. The error drastically decreases once the time step is sufficiently small. There is a relatively small gain in accuracy for reducing the time step beyond that point, as shown in Fig. 8b. It can be seen that for the smaller resolution grid ($\Delta x = 0.05m$) a smaller time step is needed to ensure stability and an accurate solution, but more accurate results are gained than the larger resolution grid once a sufficiently small time step is reached. Overall it can be seen that this method will give accurate results, provided a small enough time step to ensure stability for the grid size is used.

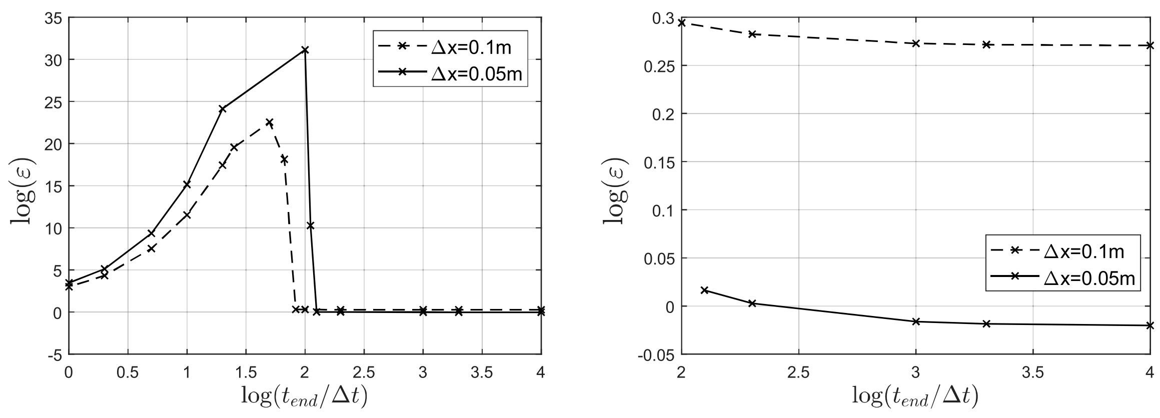
4.2.2 Implicit Time Scheme

The Heat Equation subject to these boundary and initial conditions was also solved using an implicit scheme. For this the time was treated in the same way as the spatial coordinates, so each point had coordinates of x, y and t. While coding this was conceptually simple, as the method used was the same as for the Poisson Equation, issues arose due to the large number of points, as the spatial coordinates



(a) t=0.05 (b) t=0.5 (c) t=1

Figure 7: Solution at 3 time steps for $\Delta x = 0.05m$ and $\Delta t = 0.005s$, obtained through the explicit time scheme

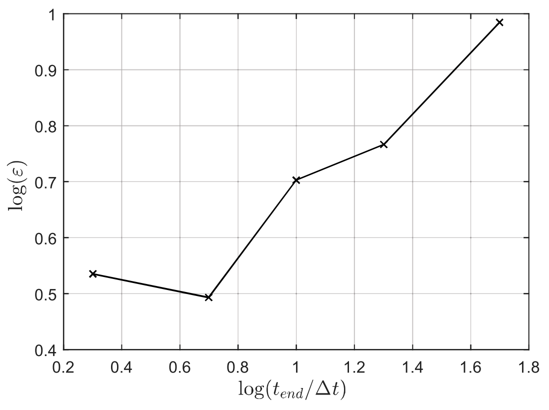


(a) (b)

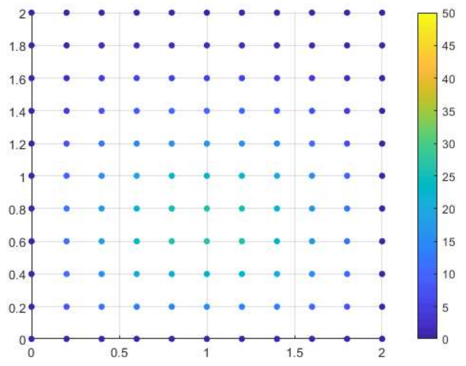
Figure 8: Graph of log(L_∞) vs. log($t_{end}/\Delta t$) for grid spacing's of $\Delta x = 0.1m$ and $\Delta x = 0.05m$, obtained through the explicit time scheme

are repeated for every Δt. Because of this attempts to use the Kansa method were unsuccessful, as the matrices created were very ill-conditioned and could not be solved accurately. Therefore a RBF-FD method was used, with a stencil size of $k = N/6$ where N is the total number of points. Additionally, the `gmres` solver in Matlab was used to solve the matrix equations as this could deal with ill-conditioned matrices better than the `mldivide` function previously used. A larger grid spacing of $\Delta x = 0.2m$ was also used in order to reduce the matrix sizes. Fig. 9a shows how the error, L_∞, varies with changing time step while Fig. 9b shows the result given by a time step of $\Delta t = 0.2s$. It can be seen that the implicit time scheme does not have the same issues in stability as the explicit time scheme, and can produce accurate results from relatively large time steps. It can also be seen that the results were ran over a smaller range of time steps than the explicit scheme and the error increases with reducing time step size. This is again because of issues with the matrices being dealt with being very large and increasingly more ill-conditioned, making them computationally expensive to solve and more likely to introduce errors in the results. For the time steps $\Delta t = 0.02s$ and $\Delta t = 0.2s$ the 'A' Matrix (as given by Eqn. 14) for the final stencil computed has very high condition numbers of 1.7×10^{35} and 7.7×10^{33} respectively.

Overall the main issue with this method is the large matrix sizes needed to represent the points in space at every time step, and the conditioning problems



(a)



(b)

Figure 9: (a) Graph of log(L_∞) vs. log($t_{end}/\Delta t$) for $\Delta x = 0.2m$ and (b) Results at $t_{end} = 1s$ with $\Delta t = 0.2s$, obtained using an implicit time scheme.

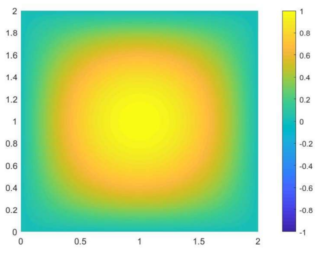
these create. It has been shown to produce relatively accurate results for large time steps that would result in instability if using the explicit time scheme. However if using a sufficiently small time step the explicit scheme will give more accurate solutions for this case.

4.3 Linear PDEs: the Wave Equation

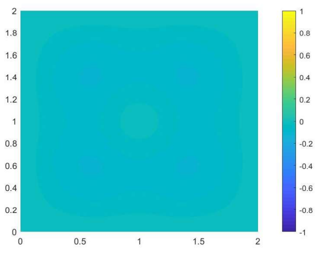
An example problem governed by the wave equation is set up such that a 2D plate with dimensions $2 \times 2m$, with a wave speed $c = 6m/s$ has Dirichlet boundary conditions, with the edges held at zero, and an initial deformation $u(x, y, 0) = xy(2-x)(2-y)$. It can be shown through separation of variables that the analytical solution to this problem, at position (x, y) and time t is:

$$u(x,y,t) = \frac{256}{\pi^6}\sum_{n=1}^{\infty}\sum_{m=1}^{\infty}\sin(\frac{m\pi}{2}x)\sin(\frac{n\pi}{2}y)(\frac{(1+(-1)^{m+1})(1+(-1)^{n+1})}{m^3n^3})\cos(3\pi t\sqrt{m^2+n^2}) \tag{69}$$

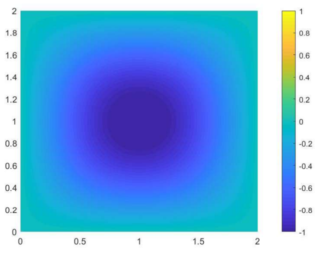
[Homer *et al.* (2017)]. Eqn. (69) was used to plot the solution to the problem at three different time steps over $t_{end} = 0.25s$ (Fig. 10).



(a) $t = 0s$



(b) $t = 0.125s$



(c) $t = 0.25s$

Figure 10: Analytical solution to the wave equation at 3 time steps for given initial and boundary conditions.

4.3.1 Explicit Time Scheme

Using the time discretisation scheme outlined in Section 3.1 a numerical solution was evaluated. This was compared to the previously obtained analytical solution using the L_∞ norm. This simulation was ran multiple times using a square grid with equal node spacing. This was done with values of $\Delta x = 0.1m$ and $\Delta x = 0.05m$ for a range of time steps Δt, as shown in Fig. 11. As expected, for large time step, the explicit scheme is unstable which leads to high error. The error increases despite decreasing values of Δt until stability is attained. Stability is reached at a larger value of Δt for a node spacing $\Delta x = 0.1m$ than for $\Delta x = 0.05m$. This is due to the inverse relationship that nodal spacing has with stability. However, once the solutions converge, the finer mesh yields more accurate results.

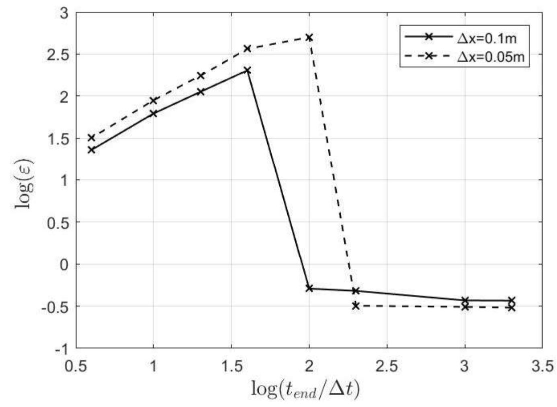


Figure 11: Plot of the logarithm of the L_∞ norm against time step Δt, for two spatial mesh sizes Δx given by an explicit scheme.

4.3.2 Implicit Time Scheme

A different approach to obtain a numerical solution is to implement time as another dimension in the RBF, and solve the three dimensional problem using Kansa's method detailed in Section 2.2.1. This can however come at an increased computational cost as the number of nodes is raised to an extra power. To improve the condition number of the matrices, the RBF-FD method is used rather than a global RBF system. The selection criteria for local stencils was a radial based approach in which nodes within a radius of $0.5m$ were selected. Since the time dimension is implemented in the RBF, time was considered to be a spatial dimension for this selection process. Contrary to the explicit scheme, the error starts decreasing with Δt until it converges. The more accurate results are also obtained by using a finer mesh. It can be noticed that at very low values of Δt, the norm increases again. As the mesh is refined, the matrix becomes ill-conditioned, increasing computational errors.

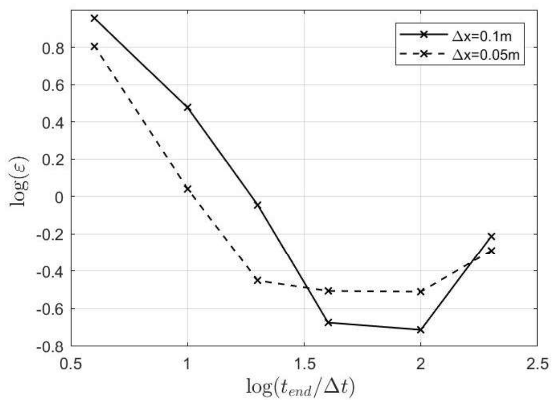


Figure 12: Plot of the logarithm of the L_∞ norm against time step Δt, for two spatial mesh sizes Δx given by an implicit scheme.

4.4 Function Reconstruction

4.4.1 3D Surface Reconstruction

RBF interpolation, HRBF implicits and Poisson Surface Reconstruction are implemented to reconstruct a sphere using 6 points. Evenly distributed 3D sample grid points, N are set the same for each algorithm for comparison. To compute the errors between all the algorithms and the analytical solutions a sphere is constructed by the following equation:

$$(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 = r^2 \tag{70}$$

where x_0, y_o, z_o is the origin of the sphere and r is its radius.

The L_2 norm (Eqn. (65)) is calculated between the sphere surface and the analytical sphere surface coordinates, so that the error is: $\epsilon = \frac{L_2}{N}$, where N is the number of sample grid points. Along with the Euclidean distance error, the time consumed for each algorithm at different sample grids numbers is recorded to demonstrate their computational performance.

Fig. 13 shows RBF interpolation and the HRBF method perform very similarly at sample points of $N = 400$, though HRBF generates less error around the body. Poisson reconstruction produces more accurate results since fewer error spots exist around the sphere.

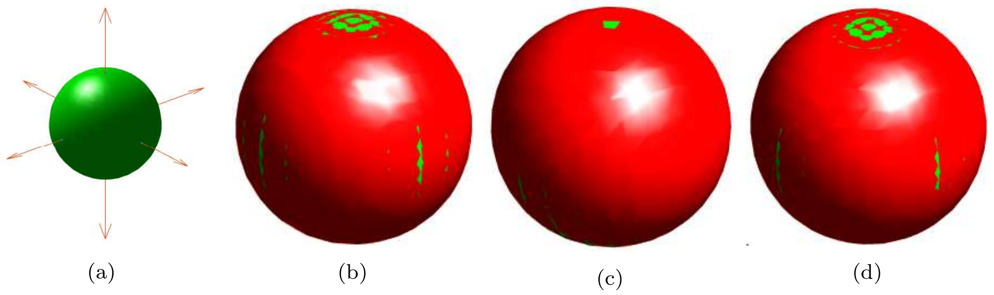


Figure 13: (a) Ground Truth. Ground truth sphere plotted on top of spheres reconstructed using: (b) RBF implicits Reconstruction, (c) Poisson Reconstruction and (d) HRBF implicits Reconstruction

Fig. 14a shows how the the error varies when sample points increase. The RBF implicits interpolation solver has not shown variation of error when using more sample points. However, the Poisson solver shows an increase of error when sample points used are greater than $N = 15^3$. Error decreases when using more sample points with applying the HRBF method. Experiments using more points are not examined for HRBF and the Poisson method since Fig. 14b has shown that significant computing power consumption occurs when using $N = 30^3$ points. The RBF implicit interpolation method is computationally economic but less accurate and smooth.

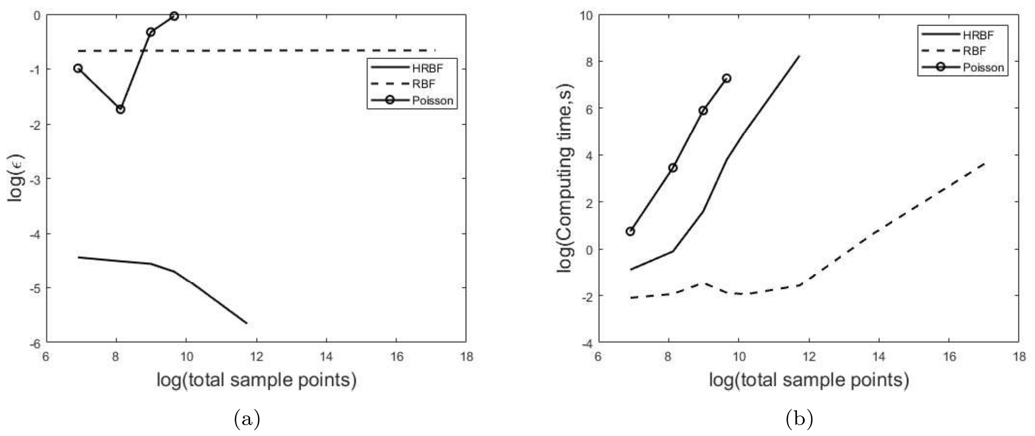


Figure 14: (a) log(ϵ) vs. log(total sample points), (b) log(computing time) vs. log(total sample points)

4.4.2 2D Boundary Reconstruction

Another application using image reconstruction methods is to reconstruct 2D boundaries. The boundary can be reconstructed using very few points, and reconstruction also removes the majority of the noise. However, since the sample points are randomly selected, there can be a disadvantage of missing details.

As shown in Fig. 15, a terrain map is segmented using the Canny technique mentioned in Section 3.2. 80 points are randomly selected from the highest-gradient-change pixels, which most likely lie on the edge of the lake. High intensity noise has a high gradient change too, and is removed in the Hysteresis Thresholding stage. A distance constraint is applied when selecting the random points so that only points that are greater than certain distances between each adjacent pair can be selected. Points selected have a certain distance between them, so that they are more uniformly distributed in the grid. This prevents the selected points from being clustered and from a result bias, from higher weighting in a particular area.

Additionally, the Poisson Surface Reconstruction technique requires interpolating the divergence of the vector field globally. Hence, the sample grid needed to be wide enough to cover the data points. In other words, the data points can not be placed on the boundaries of the sample grid. In practice, the sample grids are set twice as wide as the data point range for all algorithms.

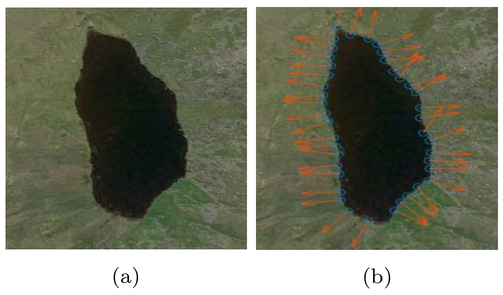


Figure 15: (a) Lake, (b) Edge segmentation

As shown in Fig. 15, the edge detection algorithm has performed well in terms of filtering noise and selecting essential data points. Based on the same data points, the reconstruction of the lake by the three techniques is illustrated in Fig. 16. All algorithms have reconstructed a connected outline of the lake. RBF interpolation shows the greatest detail, by having the sharpest edges. Poisson reconstruction tends to soften the edges using the same number of points. The HRBF implicits method keeps details of the outline but removes sharp corners [Macêdo *et al.* (2011)].

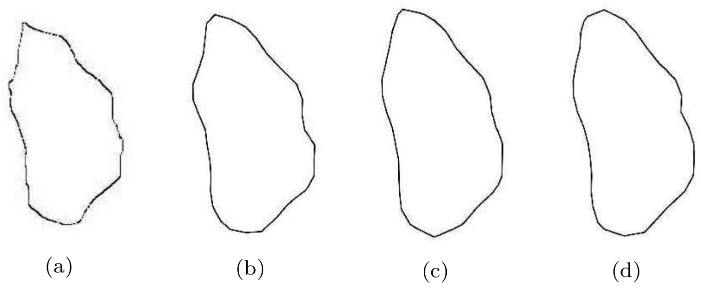
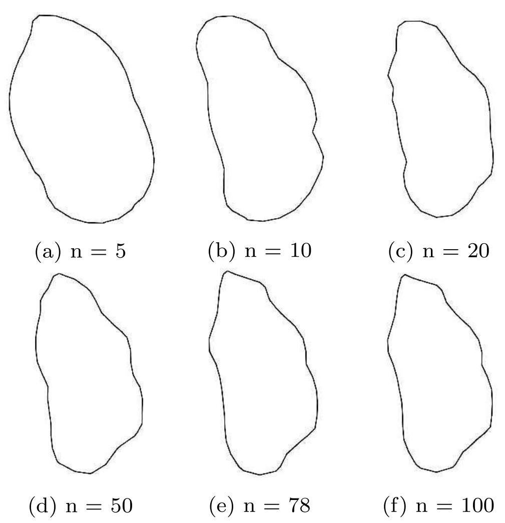


Figure 16: (a) Ground Truth, (b) RBF implicit Reconstruction, (c) Poisson Reconstruction and (d) HRBF implicit Reconstruction

An experiment using a different number of input data points is done by employing the HRBF based solver. As stated previously, a distance restriction is applied to the input points avoiding points that are too close. Fig. 17 depicts how the shape morphs with different amounts of input data points. The sequence of reconstruction varies from a rough circular shape to a more detailed outline of the original image. Regardless if more points are added as shown in Fig. 17f, the shape does no longer give further details of the lake. That is because the distance restriction has limited the minimum Euclidean distance between points. The distance restriction also limits the maximum number of points that can be selected at one time. In theory, the smaller the distance restriction, the more details can be obtained, but possibly bringing a higher degree of bias too. Another key result proving that the HRBF method can also handle coarse and non-uniform data robustly is shown in Fig. 17a.


(a) n = 5 (b) n = 10 (c) n = 20

(d) n = 50 (e) n = 78 (f) n = 100

Figure 17: 2D Boundary Reconstruction using different sets of sample points

4.5 Non linear PDEs: Burgers' Equation

4.5.1 Non linear least squares methods

The Gauss-Newton and the Levenberg-Marquadt method have been used for the solution of the Burgers' equation. The two methods have been assessed and compared for their level of accuracy and convergence, using the local RBF-FD collocation method.

For both 1D and 2D, the Gauss-Newton algorithm has shown a superior performance compared to the Levenberg-Marquadt algorithm. Using the same spatial and temporal parameters and stopping criteria, the maximum absolute error was notably smaller using a Gauss-Newton algorithm. The maximum absolute errors obtained from both methods are given in Tab. 1. It can be seen that the Gauss-

Table 1: Comparison of the L_∞ norm for the Gauss-Newton and Levenberg-Marquadt methods.

Maximum absolute error (L_∞)	Gauss-Newton	Levenberg-Marquadt
1-D	2.124×10^{-4}	9.0717×10^{-4}
2-D		
$L_\infty(\hat{\boldsymbol{u}}_x - \boldsymbol{u}_x)$	1.5×10^{-3}	1.7×10^{-3}
$L_\infty(\hat{\boldsymbol{u}}_y - \boldsymbol{u}_y)$	2.5×10^{-3}	2.7×10^{-3}

Newton algorithm provides better results in both 1D and 2D equations, although the difference in error between the two errors becomes remarkably lower for the two-dimensional case. Moreover, the G-N method also showed a faster convergence than Levenberg-Marquadt in all cases. For a stopping criteria set at $||f_{gn}(x)||_\infty$ or $||g_{lm}(x)||_\infty \leq 10^{-5}$, respectively and for the 1D case, the Gauss-Newton method was observed to converge at the 2nd iteration whilst the Levenberg-Marquadt algorithm converged at the 13th iteration. The convergence of both algorithms, including the damping parameter μ of the L-M method is presented in Fig. 18.

The Levenberg-Marquadt method takes longer to converge and this is due to the trust-region strategy where the norm of the step is limited. As mentioned in Section 3.3, very large steps are restricted. The damping parater μ, which controls the

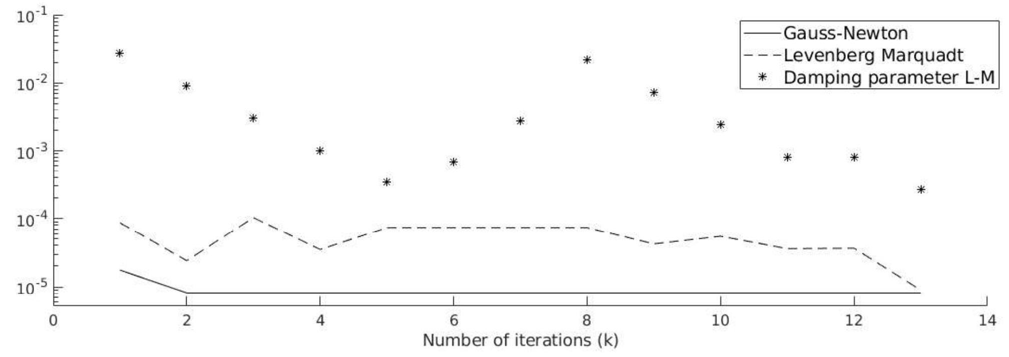


Figure 18: Convergence of non linear least squares methods for the 1D Burgers' equation.

trust region size, geometrically adds a paraboloid centered at the descent direction $h_{lm} = 0$, which results in a smaller step. Furthermore, an increase of the damping parameter results, in essence, to a step direction closer to the steepest descent method, which has slower convergence and generally results in poorer performances. For problems, like the ones considered in this research, where the value of the gain ratio ϱ, described in Eqn. 42, which controls the size of the damping parameter μ, is large and positive, the pure Gauss-Newton method is already good enough for the solution of the non linear PDE.

Therefore, due to a smaller error and a faster convergence, the Gauss-Newton method was observed to provide better results than the L-M algorithm. For this reason, the Gauss-Newton method is the non linear least squares method that has been used in all implicit schemes for the 1D and 2D Burgers' equation.

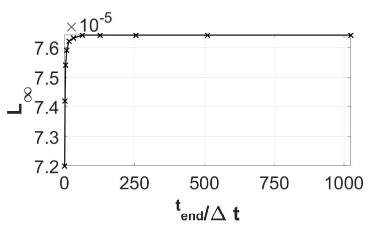
4.5.2 1D Burgers' Equation

The 1D Burgers' equation is solved numerically using the schemes described in Section 3.4. The equation is solved in the domain $x \in (0, 1)$ with a spatial discretisation of $\Delta x = 0.1$, hence $N = 11$ points are used. Moreover, $k = 3$ is used for the explicit local scheme domain of influence (Section 3.4.2) and is the number of neighbours used for both the RBF-FD schemes (Sections 3.4.3, 3.4.5). The Reynolds number used is $Re = 100$ and the solution is provided at $t_{end} = 1$. These parameters are used to provide consistent convergence studies for all five schemes and, furthermore, they are in accordance with data given on the numerical solution of the 1D Burgers' equation by Ali *et al.* (2011) and Khater *et al.* (2008), who used a meshless method of lines (MOL-RBF) and the Chebyshev spectral collocation (ChSC) method respectively.

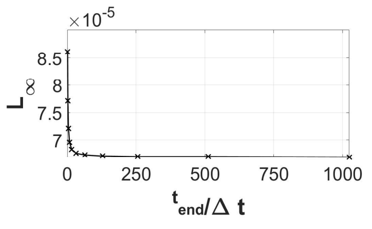
Convergence studies for all schemes are shown in Fig. 19, where the L_∞ (Eqn. (65)) is used to quantify the accuracy of the results. In the convergence studies, the time discretisation is altered rather than the spatial discretisation, since more nodes in the spacial domain would lead to an ill-conditioned system to be solved in the case of global RBF based methods where the whole matrix is populated. The time step Δt is varied between 0.25 and $\frac{0.25}{1024} = 0.00024$. The plots depict the number of iterations as an increase in a normalised time variable $(\frac{t_{end}}{\Delta t})$. The specific solutions for which the absolute error is at a minimum are given in Tab. 2 along with the time step at which this occurs.

A thorough analysis of the graphs shows that the global explicit scheme (Fig. 19a) as well as the implicit schemes (Figs. 19c, 19e) show an error increase in the numerical results as the time step is increased. This is because all three schemes have a large condition number (Tab. 2) of either the discretisation matrix or the Jacobian, used to solve a linear system at every time iteration, or iteration of the Gauss-Newton method. The condition numbers provided are the largest ones observed from all iterations (for the implicit schemes) and at the time step (Δt) noted on Tab. 2. Large condition numbers lead to errors which accumulate from every time step. An increase in error following a decrease in time step when numerically solving the 1D Burgers' equation was also documented by Ali *et al.* (2011). Conversely, the explicit local RBF (EL-RBF), and explicit RBF-FD (ERBF-FD) schemes (Figs. 19b, 19d), result in well conditioned matrices which arise from the local nature of the schemes discussed in Section 3.4.

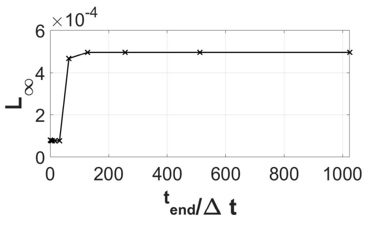
A decrease in error, noted in the the EL-RBF, ERBF-FD schemes is expected for an explicit Euler scheme which is first order in time ($\tau = O(\Delta t)$), and the truncation error is expected to decrease following an increase in Δt. The final existing error is also a result of the first order explicit scheme in time. Accuracy can be increased by creating a scheme that is of higher order in time.



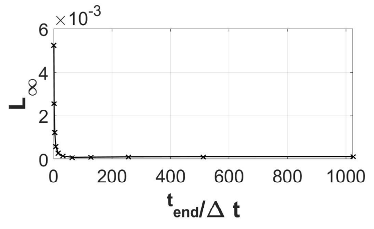
(a)



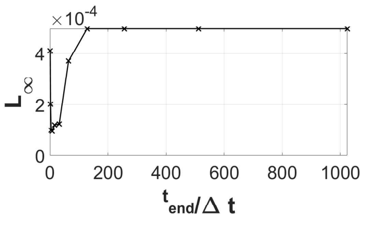
(b)



(c)



(d)



(e)

Figure 19: Convergence studies for the 1D Burgers' equation solved using (a) Global RBF discretisation in space and explicit Euler in time (EG-RBF), (b) Local RBF discretisation in space and explicit Euler in time (EL-RBF), (c) Implicit Euler discetisation in time (IG-RBF), (d) Explicit Euler in time and RBF-FD in space (ERBF-FD), (e) Implicit Euler in time and RBF-FD in space (IRBF-FD). $t_{end} = 0.25$ is the total running time of the simulation and Δt, the time step. The spatial discretisation used is $\Delta x = 0.1$.

Table 2: Lowest error norms from numerical results of all the discretisation schemes along with the condition number of the discretisation matrix A, ${}_jA$ for the explicit schemes and J^TJ for the implicit schemes. $t_{end} = 0.25, Re = 100$ and $N = 11$.

	EG-RBF (Sec. 3.4.1)	EL-RBF (Sec. 3.4.2)	ERBF-FD (Sec. 3.4.3)	IG-RBF (Sec. 3.4.4)	IRBF-FD (Sec. 3.4.5)
$\Delta t(s)$	$\frac{1}{4}$	$\frac{1}{1024}$	$\frac{1}{256}$	$\frac{1}{128}$	$\frac{1}{32}$
L_∞	7.2^{-5}	6.7×10^{-5}	7.3×10^{-5}	7.7×10^{-5}	9.5×10^{-5}
$\kappa(A), \kappa(J^TJ)$	2.16×10^9	9.98×10^3	9.98×10^3	3.98×10^{18}	3.43×10^{18}

All schemes give similarly accurate results since the L_∞ errors are within 30% of each other. The lowest error is given by the explicit local RBF scheme (EL-RBF) which uses Kansa's method to locally discretise the spatial derivatives along with an explicit time discretisation. The local scheme creates a well conditioned system which is solved at every iteration. It does not create large errors which accumulate and results are slightly better in comparison to other schemes. The explicit local RBF scheme (EL-RBF) is compared to other studies. Tab. 3, includes error norms (L_∞) at different times (t_{end}) and compares the results from the EL-RBF scheme to a Chebyshev spectral collocation (ChSC) method taken from Khater *et al.* (2008) and a method of lines with multiquadrics (MOL-MQ) scheme used by Ali *et al.* (2011). Tab. 3 shows that the EL-RBF method gives accurate results at low *Re* numbers but at higher t_{end}. It is suggested that this might be because the system is well-conditioned and there are no large error accumulations.

Table 3: Error norms (L_∞) for different times and at different Reynolds numbers for three different numerical schemes, all using $N = 11$ and $\Delta t = 0.01$. Comparison results are using the ChSC scheme [Khater *et al.* (2008)] and MOL-MQ scheme [Ali *et al.* (2011)]. Results from this present work are in bold.

	$t_{end} = 0.1$			$t_{end} = 0.25$		
Re	**EL-RBF**	MOL-MQ	ChSC	**EL-RBF**	MOL-MQ	ChSC
100	$\mathbf{3.30 \times 10^{-5}}$	3.07×10^{-5}	3.06×10^{-5}	$\mathbf{6.60 \times 10^{-5}}$	7.65×10^{-5}	7.62×10^{-5}
1000	$\mathbf{6.01 \times 10^{-7}}$	3.06×10^{-7}	3.06×10^{-7}	$\mathbf{1.13 \times 10^{-6}}$	7.66×10^{-7}	7.82×10^{-7}
10000	$\mathbf{1.71 \times 10^{-7}}$	1.71×10^{-8}	2.24×10^{-8}	$\mathbf{8.26 \times 10^{-7}}$	4.89×10^{-8}	8.94×10^{-8}

An example of the solution, which follows the *tanh* function (Eqn. (46)), is shown in Fig. 20. A larger final time is used to show the propagation of velocity in the domain (from left to right).

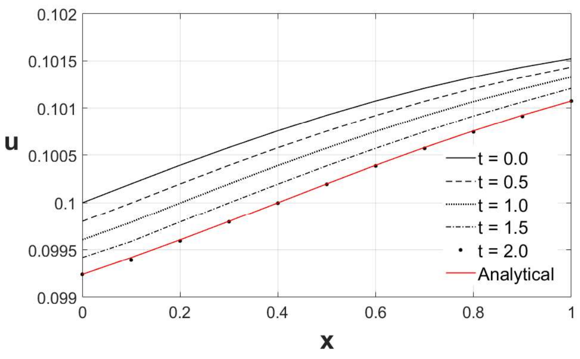


Figure 20: Numerical Solution of the 1D Burgers' equation at different time steps using the Implicit Global RBF scheme (IG-RBF, Section 3.4.1) plotted with the final analytical solution of the equation. Parameters used are $Re = 1000$, $\Delta x = 0.05$, $\Delta t = 0.05$ and $t_{end} = 4$.

4.5.3 2D Burgers' Equation

In the case of the 2D Burgers' equation solution, to accurately represent the domain, the domain size chosen in each direction is $N_x = N_y = 21$ and a total of $N = 441$ points overall. The domain also being $x \in (0, 1), y \in (0, 1)$ and hence the spacial discretisation is $\Delta x = 0.05, \Delta y = 0.05$. The local domain of influence for the local RBF scheme (Section 3.4.2) and the number of neighbours for the RBF-FD schemes (Sections 3.4.3, 3.4.5) is $k = 5$. The flow Reynolds number used for the convergence studies for all schemes is $Re = 60$ as the explicit schemes become unstable at $Re > 100$, where the solution becomes convection dominated [Zhang (2009)]. Nevertheless, the most accurate scheme is then compared to analytical results and the numerical results given by Sarler *et al.* (2012) and Bahadır (2003) where a higher Reynolds number is used ($Re = 100$). The total time is set to $t_{end} = 2$ as in the aforementioned studies.

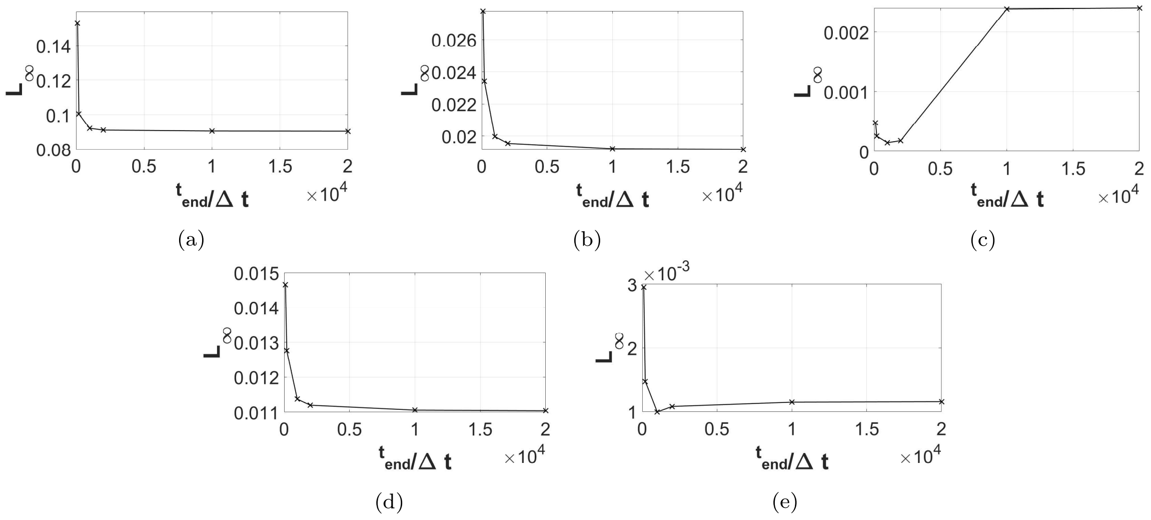


Figure 21: Convergence studies for the 2D Burgers' equation solved using (a) Global RBF discretisation in space and explicit Euler in time (EG-RBF), (b) Local RBF discretisation in space and explicit Euler in time (EL-RBF), (c) Implicit Euler discetisation in time (IG-RBF), (d) Explicit Euler in time and RBF-FD in space (ERBF-FD), (e) Implicit Euler in time and RBF-FD in space (IRBF-FD). $t_{end} = 2$ is the total running time of the simulation and Δt, the time step. The error norm (L_∞) is calculated using the velocity in the x direction, u_x. The spatial discretisation used is $\Delta x = 0.05$.

Fig. 21 depicts the convergence studies of the infinity norm with a normalised measure for time, which is the total number of iterations in time. The number of iterations used lies from 100, where a solution is provided by all schemes, up to 20,000 iterations where all schemes converge. The explicit time schemes (Figs. 21a, 21b, 21d) provide results that converge at $\frac{t_{end}}{\Delta t} = 20000, \Delta t = 0.0001$ and the schemes that are implicit in time (Figs. 21c, 21d) converge at $\frac{t_{end}}{\Delta t} = 1000, \Delta t = 0.002$.

Results from implicit schemes start becoming less accurate at a certain time step and the maximum error, L_∞, starts to increase. For this reason, the most accurate result is obtained at a larger time step, in contrast to explicit schemes. The reasons for this lie in the underlying coding and solution procedure for the different methods. The explicit schemes all iterate in time using information from previous time steps (Eqn. 50), hence this requires just one linear system to be solved at every time iteration (Sections 3.4.1, 3.4.2,3.4.3) . The linear system for the EG-RBF, IG-RBF and IRBF-FD schemes are solved using `gmres` in MATLAB, since the matrices used to solve the linear systems (A, JJ^T) have high condition numbers ($\approx 10^{18}$). The solver is used at a desired tolerance of $\tau = 10^{-6}$, hence there is an underlying error that accumulates. Accumulation is more apparent in the implicit schemes at higher iterations since the linear system (Eqn. 39) is solved for a k_{max} number of iterations at every time step, until it converges to a solution of the velocity for the next time step (u_x^{n+1}, u_y^{n+1}). Additionally, another error that accumulates every time step from the implicit solvers is $\|f\|_\infty < 10^{-5}$. This error results from solving the non linear least squares problem at every iteration and the final result will be comparatively less accurate as simulation time increases.

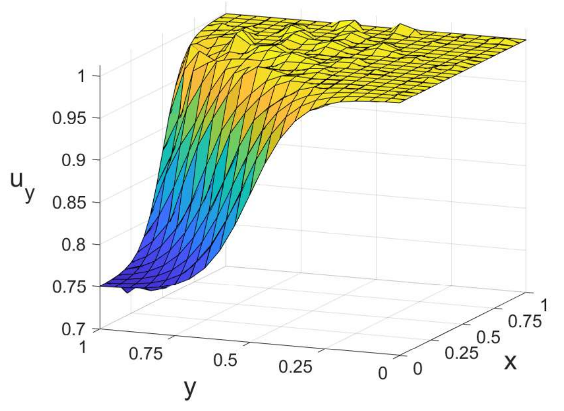
Table 4: Lowest error norms from numerical results of all the discretisation schemes where $t_{end} = 2, Re = 60$ and $N = 441$.

	EG-RBF (Sec. 3.4.1)	EL-RBF (Sec. 3.4.2)	ERBF-FD (Sec. 3.4.3)	IG-RBF (Sec. 3.4.4)	IRBF-FD (Sec. 3.4.5)
$\Delta t(s)$	0.0001	0.0001	0.0001	0.002	0.002
$L_\infty(\hat{\boldsymbol{u}} - \boldsymbol{u}_x)$	9.0×10^{-2}	1.9×10^{-2}	1.1×10^{-2}	1.5×10^{-4}	1.0×10^{-3}
$L_\infty(\hat{\boldsymbol{u}} - \boldsymbol{u}_y)$	9.0×10^{-2}	1.4×10^{-2}	6.3×10^{-3}	1.6×10^{-4}	9.8×10^{-4}

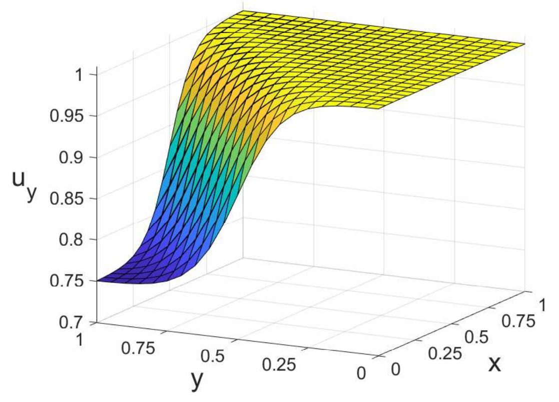
The most accurate solutions from all the schemes are provided in Tab. 4. The performance of the explicit global (EG-RBF) scheme is the poorest, which can be attributed to the large condition number of the matrix ($\kappa(A) = 3.3 \times 10^{17}$), used to find the coefficients (λ) for the next iteration, compared to the condition numbers of the local explicit (EL-RBF) and explicit RBF-FD (ERBF-FD) schemes which are designed to use well-conditioned matrices. In Sarler *et al.* (2012), it is noted that the 2D coupled Burgers' equations become convection dominated at high Reynolds numbers (where viscosity is very low), and a sharp front appears in the solution of the equations. The discontinuity causes instabilities which grow in time. This is solved by Sarler *et al.* (2012), using the LRBFCM method to discretise the equations in space with an upwinding technique and by Bahadır (2003) through an implicit scheme in both space and time. In this research, implicit time schemes are used along with the Kansa RBF and RBF-FD spacial discretisations to solve for high Re number flows. The inaccuracy of the explicit schemes due to a sharp front is noticeable even at $Re = 60$, but the implicit schemes show stable results due to their inherent stability. Between these, the RBF-FD gives more accurate results than its global RBF counterpart(at higher Re numbers), and it also has a much lower running time since operations are less computationally expensive, and higher accuracy could be achieved by increasing the number of neighbours used in the weight generation (Section 2.2.2).

In order to prove the fact that spurious oscillations occur at higher Reynolds numbers with the explicit schemes, the numerical result for u_y using the global explicit scheme (EG-RBF) at $t_{end} = 2$ is shown in Fig. 22a. The unstable oscillations come from the convection dominated equation and the fact that the explicit scheme blindly solves for the next time step using only past information. In contrast, the implicit RBF-FD scheme is used to provide numerical results using the same time step, spacial discretisation and final time. The result is shown in Fig. 22b, where no oscillations exist.

Finally, a comparison between the implicit schemes used in this study and the methods used to discretise time and space by Bahadır (2003) and Sarler *et al.* (2012) is given in Tab. 5 for u_x and Tab. 6 for u_y. Tables 5, 6 show that the implicit scheme with RBF-FD used in this study gives comparative results to the other studies with very similar errors. The reason for this could be the fact that the discretisation is of first order in time in all three studies, $O(\Delta t)$. The difference between them is



(a)



(b)

Figure 22: Numerical solutions u_y at $Re = 100$, $N = 441$, $t_{end} = 2$ and $\Delta t = 0.002$. (a) Solution using the explicit global RBF scheme (EG-RBF), (b) Solution using the implicit RBF-FD scheme (IRBF-FD).

the discretisation in space, with Sarler *et al.* (2012) discretising the spatial derivatives using a local Kansa RBF scheme (LRBFCM), Bahadır (2003) using a centred scheme and RBF-FD used in this study. For Bahadır (2003), the discretisation is second order accurate in space, $O(\Delta x^2)$. RBF discretisations are shown to perform well even compared to higher order schemes, hence the slightly more accurate results from the schemes that use RBFs to discretise the spatial derivatives [Fornberg & Flyer (2015)]. A contrasting result to this statement is that the IG-RBF scheme gives the least accurate results. The reason lies in computational errors, which arise due to the very high condition numbers of the matrices used in the global scheme ($\kappa(J^T J) = 4.6 \times 10^{19}$), where the discretisation matrix is densely populated. Moreover, the time discretisation used is lower than the optimal value for the global RBF scheme (Tab. 4). Finally, a higher accuracy could be achieved by increasing the number of points in the RBF-FD stencil ($k > 5$) before the results are affected by ill-conditioned matrices, leading to more accurate results than the other schemes presented here.

Table 5: Numerical results and L_∞ for u_x from the exact solutions and various schemes including IG-RBF and IRBF-FD from this work (in bold), LRBFCM and the fully implicit FD method from Sarler *et al.* (2012) and Bahadır (2003) respectively. Solutions given at typical mesh points with $t_{end} = 2$, $Re = 100$, $\Delta t = 0.0001$, $N = 441$.

	$\boldsymbol{u_x}$						$\boldsymbol{L_\infty}$
(x, y)	$(0.1, 0.1)$	$(0.3, 0.3)$	$(0.5, 0.5)$	$(0.3, 0.7)$	$(0.1, 0.9)$	$(0.5, 0.9)$	
Analytical	0.50048	0.50048	0.50048	0.55568	0.74426	0.55568	0
IG-RBF	**0.49998**	**0.50319**	**0.50391**	**0.55728**	**0.74791**	**0.55894**	**0.0037**
IRBF-FD	**0.50045**	**0.50037**	**0.50029**	**0.55450**	**0.74413**	**0.55407**	**0.0016**
Sarler *et al.* (2012)	0.50047	0.50044	0.50041	0.55481	0.74420	0.55568	0.0012
Bahadır (2003)	0.49983	0.49977	0.49973	0.55429	0.74340	0.55413	0.0016

Fig. 23 shows the results for the velocity components of the 2D Burgers' equation (u_x and u_y).

Table 6: Numerical results and L_∞ for u_y from the exact solutions and various schemes including IG-RBF and IRBF-FD from this work (in bold), LRBFCM and the fully implicit FD method from Sarler *et al.* (2012) and Bahadır (2003) respectively. Solutions given at typical mesh points with $t_{end} = 2$, $Re = 100$, $\Delta t = 0.0001$, $N = 441$.

	$\boldsymbol{u_y}$						$\boldsymbol{L_\infty}$
(x, y)	$(0.1, 0.1)$	$(0.3, 0.3)$	$(0.5, 0.5)$	$(0.3, 0.7)$	$(0.1, 0.9)$	$(0.5, 0.9)$	
Analytical	0.99952	0.99952	0.99952	0.94433	0.75574	0.94433	0
IG-RBF	**0.99965**	**0.99702**	**0.99626**	**0.94276**	**0.75210**	**0.94113**	**0.0036**
IRBF-FD	**0.99949**	**0.99941**	**0.99934**	**0.94512**	**0.75578**	**0.94541**	**0.0011**
Sarler *et al.* (2012)	0.99953	0.99956	0.99959	0.94520	0.75580	0.94551	0.0012
Bahadır (2003)	0.99826	0.99861	0.99821	0.94409	0.75500	0.94441	0.0014

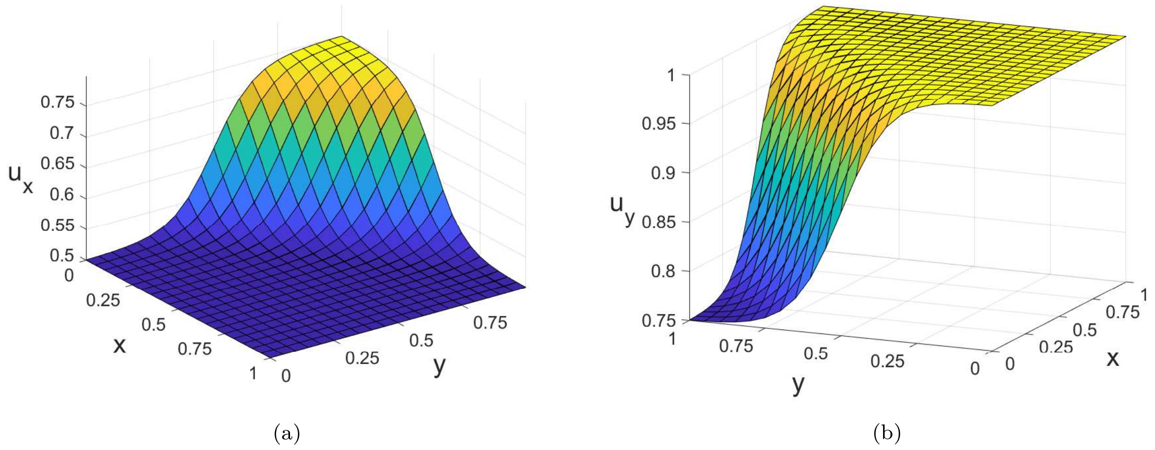


Figure 23: Numerical solutions for the 2D Burgers' equation, (a) u_x (b) u_y using IRBF-FD at $t_{end} = 2$, $Re = 100$, with $N = 441$, $\Delta t = 0.0001$.

5 Conclusions

Radial basis function based methods for the solution of partial differential equations provided accurate and computationally efficient results for all the applications considered. They have proven to be a valuable tool for interpolation and subsequently the discretisation of PDEs. Higher computational efficiency was achieved by locally supported RBF methods, such as RBF-FD and LRBFCM, as they mitigate densely populated matrices which result from the use of the global Kansa method. A numerical solution was given to linear PDEs, the Poisson, heat and wave equation; surface and boundary reconstruction applications and finally, the Burger's equation.

The Poisson Equation was solved using both Kansa and RBF-FD methods and compared to an analytical solution. Both gave accurate results. Kansa's method was tested with both a regularly spaced grid and randomly scattered points, where the accuracy of the results in both cases demonstrated the flexibility of RBF based methodologies regarding the geometry. It was found for both the heat and wave equation, an explicit time scheme gave accurate results provided a sufficiently small time step. An implicit time scheme was also implemented, which was more accurate for large time steps however had issues with the conditioning of the matrix and was generally more temperamental to implement.

The 3D Surface reconstruction of a sphere gave more accurate and smoother results when using the HRBF and Kansa's method through the Poisson reconstruction methodology in comparison with an implicit RBF interpolation solver, at the expense of a higher computational cost. This is expected due to the globally supported RBFs used for both methods. Moreover, for the 2D boundary reconstruction case, edge detection algorithms successfully filtered noise and selected essential data points. In this case, RBF implicit interpolation shows results with more details whilst Poisson reconstruction tends to soften the edges and HRBF keeps the details but removes the sharp corners. It was also found that HRBF can handle coarse and non-uniform data robustly.

Additionally, RBF based methods have proven to adequately deal with the demanding field of non linear PDEs. The solutions were compared to analytical solutions and results from alternative methods provided by previous literature. The methods developed in this work performed either comparably or better than other methodologies. Revolutionary work has been done by developing an implicit time scheme to discretise the time derivatives of the Burgers' equations, in combination with both a global Kansa and an RBF-FD scheme for the spatial derivatives. The inherently stable implicit scheme was able to successfully solve stability issues caused by the sharp front of the 2D Burger's equation. For the schemes including an implicit discretisation, non linear least squares methods were used to approximate the solution and it was found that Gauss-Newton algorithms provided more accurate results and a faster performance than the Lenvenberg-Marquadt method. The IRBF-FD scheme provided promising results to the 2D Burgers' equation.

Overall, radial basis functions have proven to be a versatile tool for function interpolation, discretisation of PDEs and surface reconstruction, demonstrating high flexibility regarding the geometry. The different RBF collocation methods are relatively easy to implement for higher spatial dimensions and showed high order accuracy, compared to other existing methodologies. However, large condition numbers have shown to be an issue for globally supported collocation methods and stability issues require further investigation. Further work could be considered especially on the local support RBF based collocation methods, and particularly the RBF-FD method, which has demonstrated to be more computationally efficient and more accurate in some cases, whilst avoiding large conditioning numbers. This can include further studies on the appropriate stencil size for high order accurate results without compromising computational time, testing an optimal shape factor c for each case specifically and a comparison of different basic functions, such as the Gaussian or thin plate splines. This work could also be further improved by increasing the spatial and temporal resolution of the numerical approximations, as well as considering higher order time schemes for an even more accurate solution to PDEs.

Therefore, RBF discretisations offer a promising tool for the solution of complex geometries in the field of geosciences, computational fluid dynamics or biology, providing a flexible and high order accurate approach to the solution of PDEs. Generation of fine meshes tends to be extremely computationally expensive and the accuracy of the results is usually highly dependent on it, however RBF collocation methods successfully avoid these issues. The use of the local RBF-FD method offers opportunities for adaptive refinement and scalability to large-scale parallel computing, particularly important for Computational Fluid Dynamics and the solution of the Navier-Stokes equation, where previous literature has only developed solvers for very simple geometries (Demirkaya *et al.* (2008)). Moreover, RBF based methods also provide accurate solutions for function reconstruction, where the method is of particular interest when applied to medical or blurred images.

Acknowledgements

The authors would like to thank Dr. Alberto Gambaruto from the University of Bristol for his supervision.

References

Ali, A. & Haq, S. 2009 A computational meshfree technique for the numerical solution of the two-dimensional coupled burgers' equations. *International Journal for Computational Methods in Engineering Science and Mechanics* **10** (5), 406–422.

Ali, A., Hussain, I. & Pakhtunkhwa, K. 2011 A numerical meshless technique for the solution of some burgers' type equations.

Bahadir, A.R. 2003 A fully implicit finite-difference scheme for two-dimensional burgers' equations. *Applied Mathematics and Computation* **137**, 131–137.

Bateman, H. 1915 Some recent researches on the motion of fluids. *Monthly Weather Review* **43** (4), 163–170.

Bernal, F. 2010 Meshless methods for singular and nonlinear elliptic pde's. Lecture at Instituto Superior Tecnico.

Canny, J.F. 1986 A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **PAMI-8**, 679–698.

Carr, J.C., Fright, W.R. & Beatson, R.K. 1997 Surface interpolation with radial basis functions for medical imaging. *IEEE Transactions on Medical Imaging* **16**, 96–107.

Carr, J. C., Beatson, R. K., Cherrie, J. B., Mitchell, T. J., Fright, W. R., McCallum, B. C. & Evans, T. R. 2001 Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, p. 67–76. New York, NY, USA: Association for Computing Machinery.

Daileda, R.C. 2012 The two dimensional heat equation. Lecture at Trinity University.

Demirkaya, G., Soh, C. & Ilegbusi, O.J. 2008 Direct solution of navier–stokes equations by radial basis functions. *Applied Mathematical Modelling* **32**, 1848–1858.

Duchon, J. 1977 Splines minimizing rotation-invariant semi-norms in sobolev spaces.

Fasshauer, G. E. 2007 *Meshfree Approximation Methods with Matlab*. World SCientific.

Fletcher, C. 1983 Generating exact solutions of the two-dimensional burgers' equation. *International Journal for Numerical Methods in Fluids* **3**, 213 – 216.

Flyer, N., Wright, G. B. & Fornberg, B. 2014 Radial basis function-generated finite differences: A mesh-free method for computational geosciences. *Handbook of Geomathematics* .

Fornberg, B. & Flyer, N. 2015 *A Primer on Radial Basis Functions with Applications to the Geosciences*. USA: Society for Industrial and Applied Mathematics.

Franke, C. & Schaback, R. 1998 Solving partial differential equations by collocation using radial basis functions. *Applied Mathematics and Computing* **93**, 73–82.

Franke, Richard 1982 Scattered data interpolation: tests of some methods.

Gary, J., Helgason R. 1970 A matrix method for ordinary differential eigenvalue problems. *Journal of Computational Physics* **5** (2), 169–187.

Hardy, R. L. 1971 Multiquadric equations of topography and other irregular surfaces.

Homer, M., Lepora, N., Campbell, C. & Barton, D. 2017 Example sheet 4: Pdes: introduction separation of variables. Engineering Math 2 Worksheet, University of Bristol.

KANSA, E. J. 1990 Multiquadrics—a scattered data approximation scheme with applications to computational fluid-dynamics—ii solutions to parabolic, hyperbolic and elliptic partial differential equations. *Computers & mathematics with applications* **19** (8-9), 147–161.

KANSA, E. J, ALDREDGE, R. C. & LING, L. 2009 Numerical simulation of two-dimensional combustion using mesh-free methods. *Engineering Analysis with Boundary Elements* **33** (7), 940–950.

KAZHDAN, M. M., BOLITHO, M. & HOPPE, H. 2006 Poisson surface reconstruction. In *SGP '06*.

KHATER, A.H., TEMSAH, R.S. & HASSAN, M.M. 2008 A chebyshev spectral collocation method for solving burgers'-type equations. *Journal of Computational and Applied Mathematics* **222** (2), 333–350.

KINDELAN, M., BERNAL, F., GONZÁLEZ-RODRÍGUEZ, P. & MOSCOSO, M. 2010 Application of the rbf meshless method to the solution of the radiative transport equation. *Journal of Computational Physics* **229**, 1897–1908.

KRIGE, D. G. 1951 A statistical approach to some mine valuation and allied problems on the witwatersrand: By dg krige. PhD thesis, University of the Witwatersrand.

LE MÉHAUTÉ, A. 1990 A finite element approach to surface reconstruction.

MACÊDO, I., GOIS, J. P. & VELHO, L. 2011 Hermite radial basis functions implicits. *Comput. Graph. Forum* **30**, 27–42.

MADSEN, K., NIELSEN, H.B. & TINGLEFF, O. 2004*a* Methods for non-linear least squares problems. Informatics and Mathematical ModellingTechnical University of Denmark, DTU, 2nd Edition.

MADSEN, K., NIELSEN, H. & TINGLEFF, O. 2004*b* Methods for non-linear least squares problems (2nd ed.) p. 60.

MAI-DUY, N. & TANNER, R.I 2005 Computing non-newtonian fluid flow with radial basis function networks. *International Journal for Numerical Methods in Fluids* **48**, 1309–1336.

MALFLIET, W. 1992 Solitary wave solutions of nonlinear wave equations. *American Journal of Physics* .

MATHWORKS 2020 mldivide.

MICCHELLI, C. A. 1984 Interpolation of scattered data: distance matrices and conditionally positive definite functions. In *Approximation theory and spline functions*, pp. 143–145. Springer.

PIRET, C. 2012 The orthogonal gradients method: A radial basis functions method for solving partial differential equations on arbitrary surfaces. *Journal of Computational Physics* **231**.

RICHARDSON, L. F. 1911 The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character* **210**, 307–357.

SARLER, UL ISLAM, SIRAJ, VERTNIK & KOSEC 2012 Radial basis function collocation method for the numerical solution of the two-dimensional transient nonlinear coupled burgers' equations. *Applied Mathematical Modelling* **36**, 1148–1160.

SARLER, B. & VERTNIK, R. 2006 Meshfree local radial basis function collocation method for diffusion problems. *Computers and Mathematics with Applications* **51**, 1269–1282.

SHU, C.AND DING, H. & YEO, K. S. 2003 Local radial basis funcion-based differential quadrature method and its application to solve two-dimensional incompressible navier-stokes equations. *Computer Methods in Applied Mechanics and Engineering* **192**, 941–954.

TOLSTYKH, A.I. & SHIROBOKOV, D. 2003 On using radial basis functions in a "finite difference mode" with applications to elasticity problems. *Computational Mechanics* **33**, 68–79.

TURK, G. & O'BRIEN, J. F. 1999 Shape transformation using variational implicit functions. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, p. 335–342. USA: ACM Press/Addison-Wesley Publishing Co.

WENDLAND, H. 1995 Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in computational Mathematics* **4** (1), 389–396.

Wong, A.S.M., Hon, Y., Li, T.-S., Chung, S. L. & Kansa, E. 1998 Multizone decomposition of time dependent problems using the multiquadric scheme. *Computers Mathematics with Applications* **37**, 23–43.

Xu, S. B. & Lu, W. X. 1988 Surface reconstruction of 3d objects in computerized tomography. *Computer Vision, Graphics, and Image Processing* **44**, 270–278.

Zhang, X. H., Ouyang J. Zhang L. 2009 Element-free characteristic galerkin method for burgers' equation. *Engineering Analysis with Boundary Elements* **33** (3), 356–362.